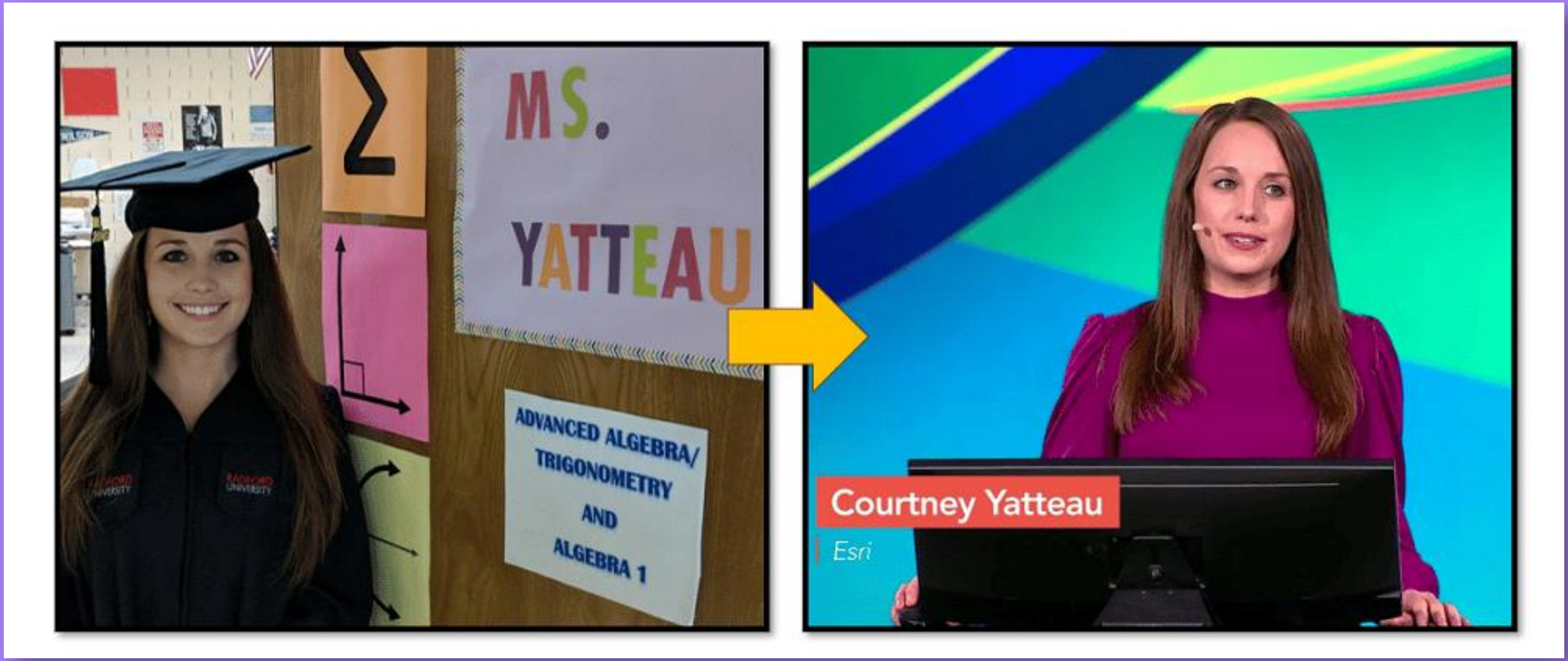
# REACT AND THE ART OF GAMIFICATION

COURTNEY YATTEAU

# COURTNEY YATTEAU

Developer Advocate, Esri



# AGENDA

01 GAMIFICATION BASICS

02 REACT OVERVIEW

03 LIVE CODING

04 G.A.M.E.S PRINCIPLES

05 REAL-WORLD EXAMPLE CONCLUSIONS

https://github.com/cyatteau/nordevcon-2025-react-gamification

GAMERS 🎮

COMPETITIVE 🔥

REWARDS 🥇

# WHAT IS GAMIFICATION?

> Gamification is the process of using **game thinking** and **game dynamics** to engage **audiences** and **solve problems**.
>
> Gabe Zichermann

# WHY GAMIFICATION?

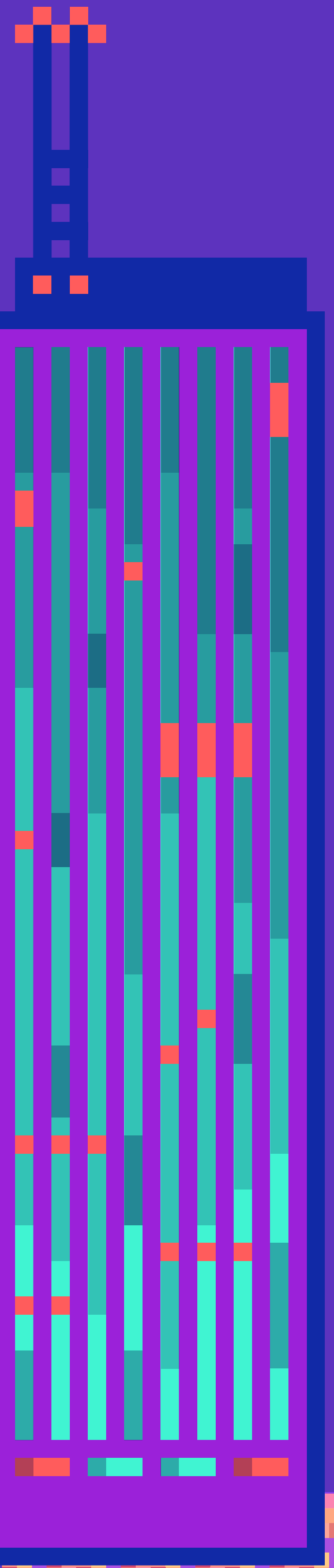
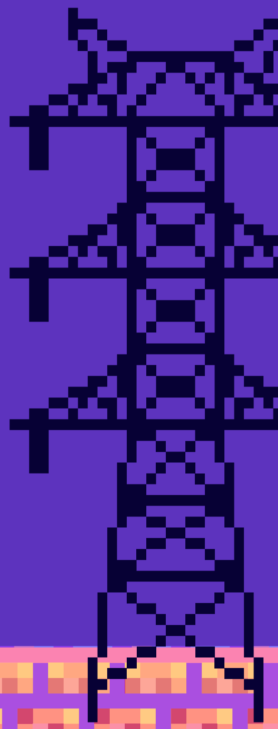
224★
Star balance
Rewards options
25
50
150
200
400
Details
Redeem
Daily Goal Exceeded!
Combo Bonus! +5
2 day streak
CONTINUE
RUN STREAK GOALS
365 DAYS
HEADSPACE
I've meditated 365 days in a row.

WHAT?

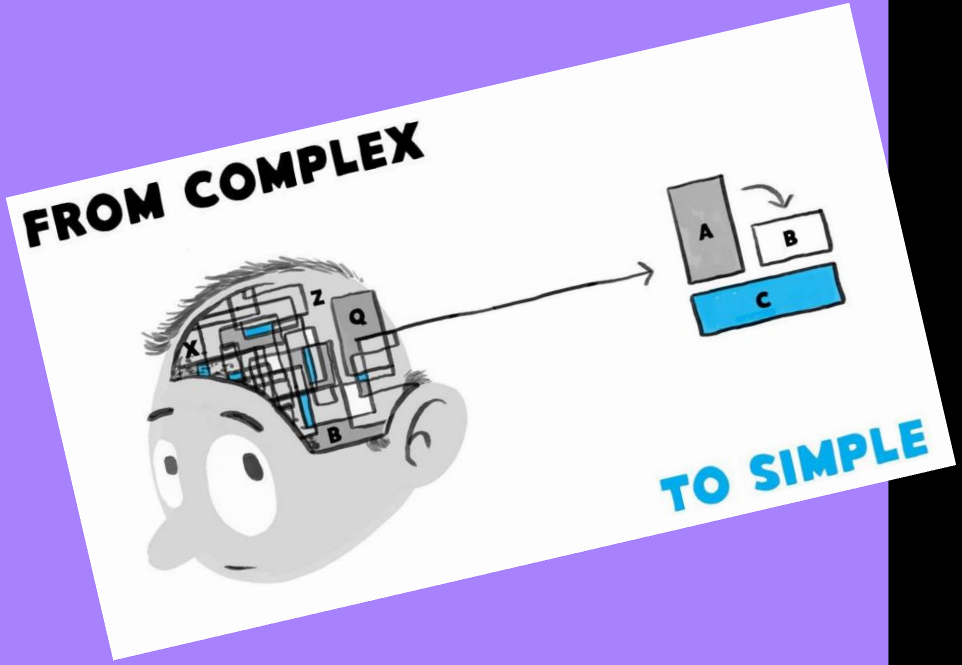
WHY?

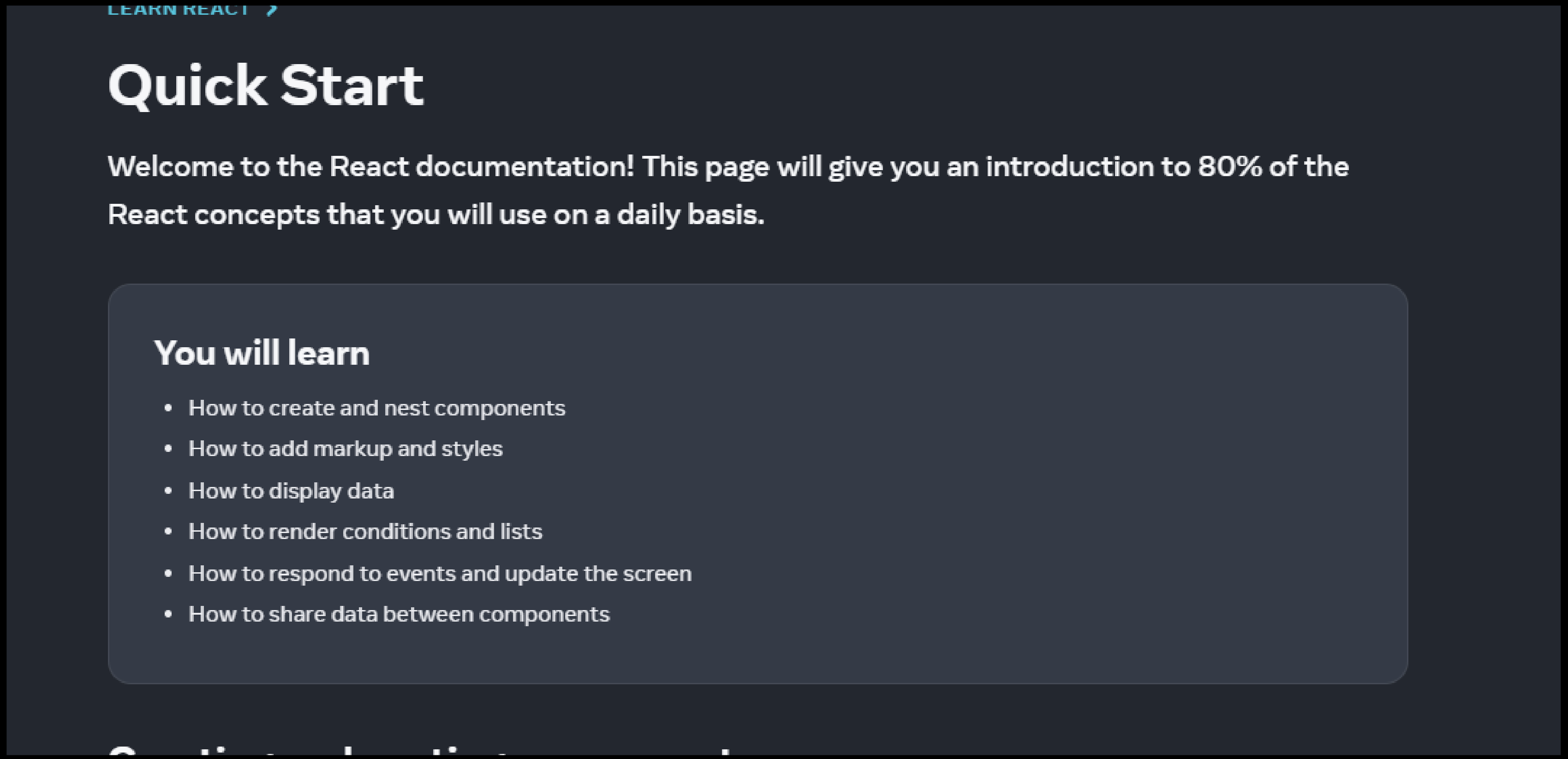
HOW?

# React JS

# GAMIFIYING REACT EDUCATION





LEARN REACT

## Quick Start

Welcome to the React documentation! This page will give you an introduction to 80% of the React concepts that you will use on a daily basis.

### You will learn

- How to create and nest components
- How to add markup and styles
- How to display data
- How to render conditions and lists
- How to respond to events and update the screen
- How to share data between components

react.dev/learn

# WHAT IS REACT?

⚛ JS LIBRARY FOR UI ⚛

Declarative 📝

Component-based 🧩

Efficient 🔄

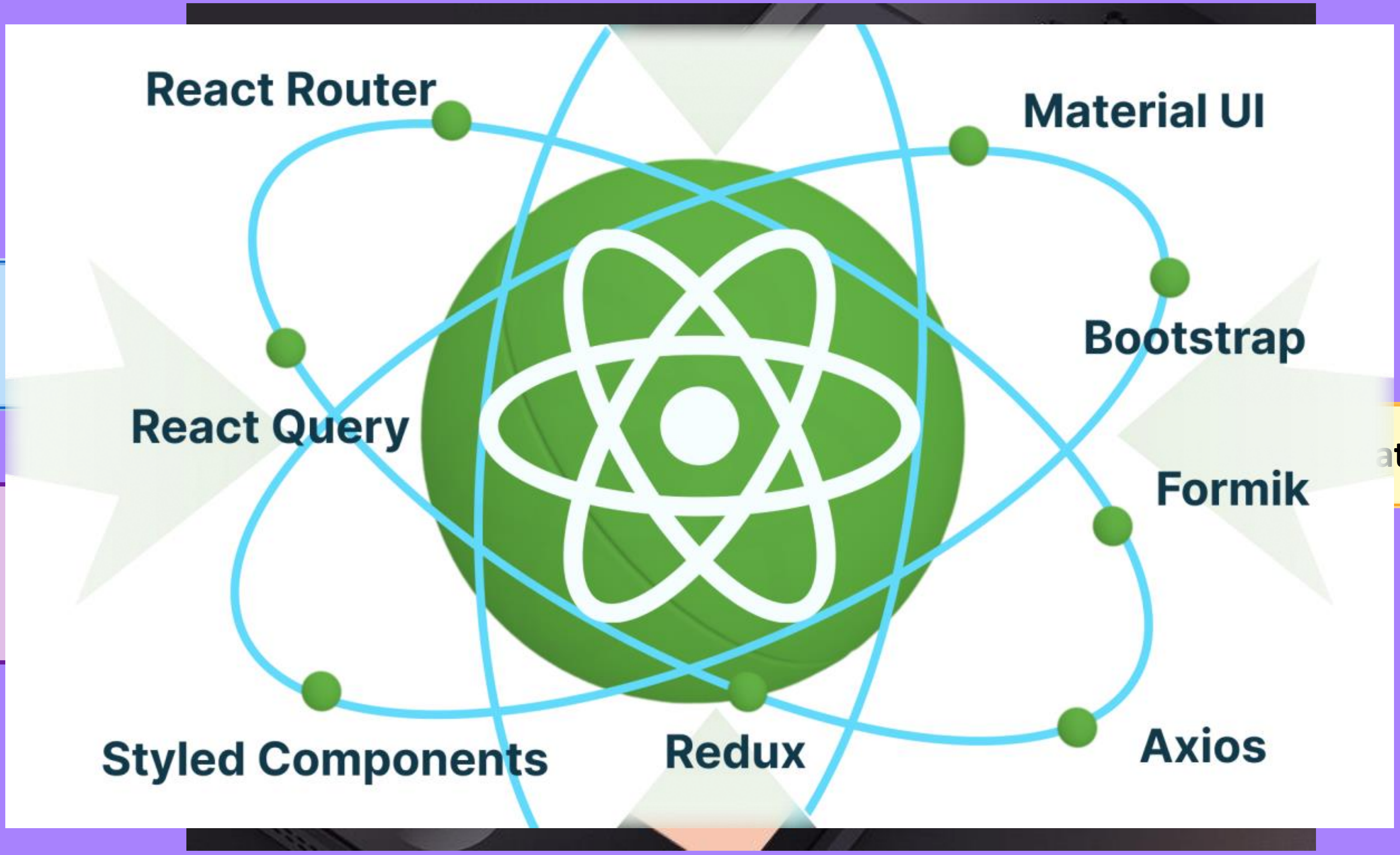
# WHY USE REACT?

## Rich Ecosystem

User Action
(e.g., click/tap)

Basic State (useState)

ate



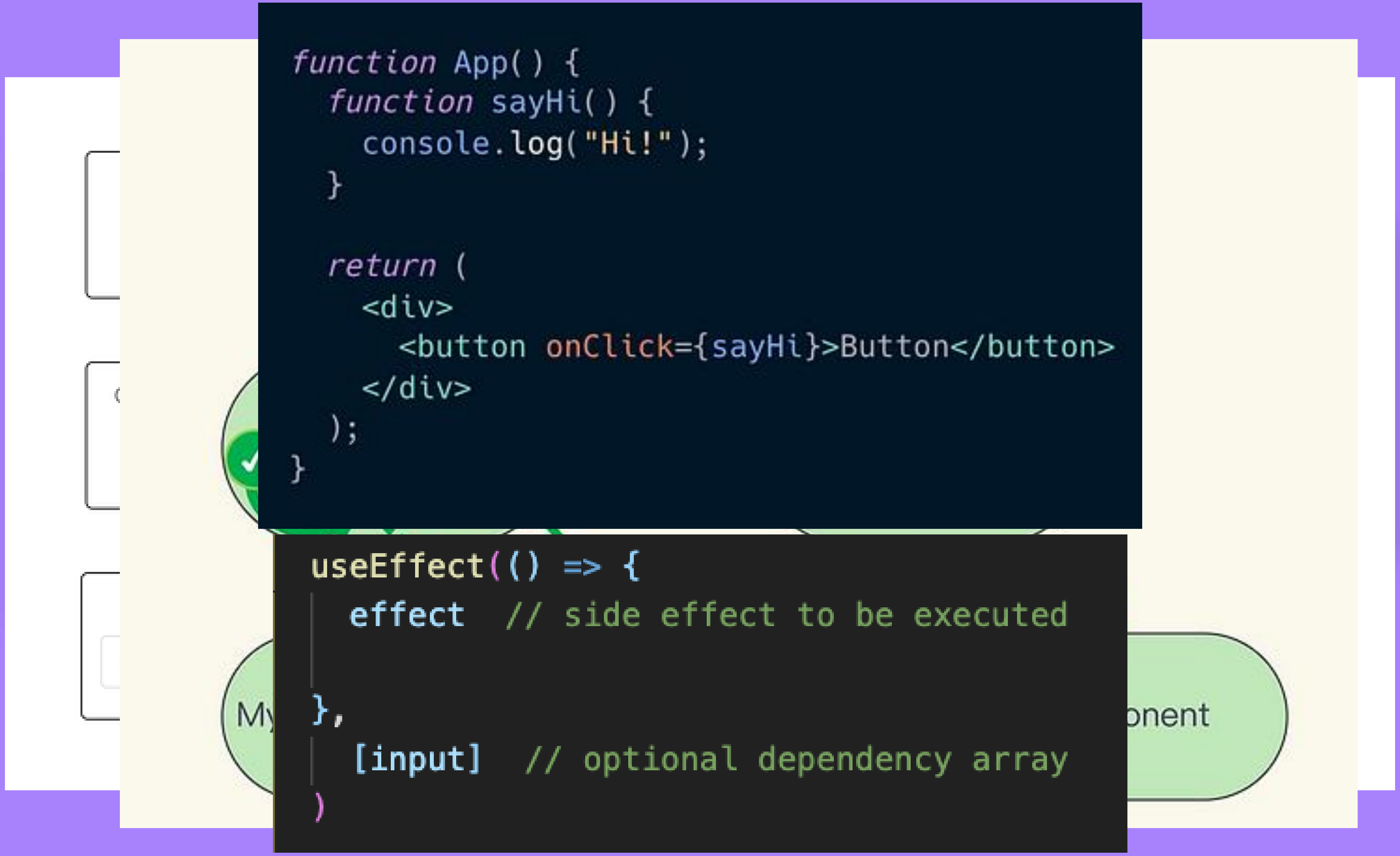
# HOW TO USE REACT?

Project Setup (Vite.js, Next.js, etc.)

Build with JSX & Components

Manage Data with State & Props

Handle Events & Effects



```
function App() {
  function sayHi() {
    console.log("Hi!");
  }

  return (
    <div>
      <button onClick={sayHi}>Button</button>
    </div>
  );
}
```

```
useEffect(() => {
  effect  // side effect to be executed

},
  [input]  // optional dependency array
)
```

# G.A.M.E.S.

G

Gamified UI Elements

A

M

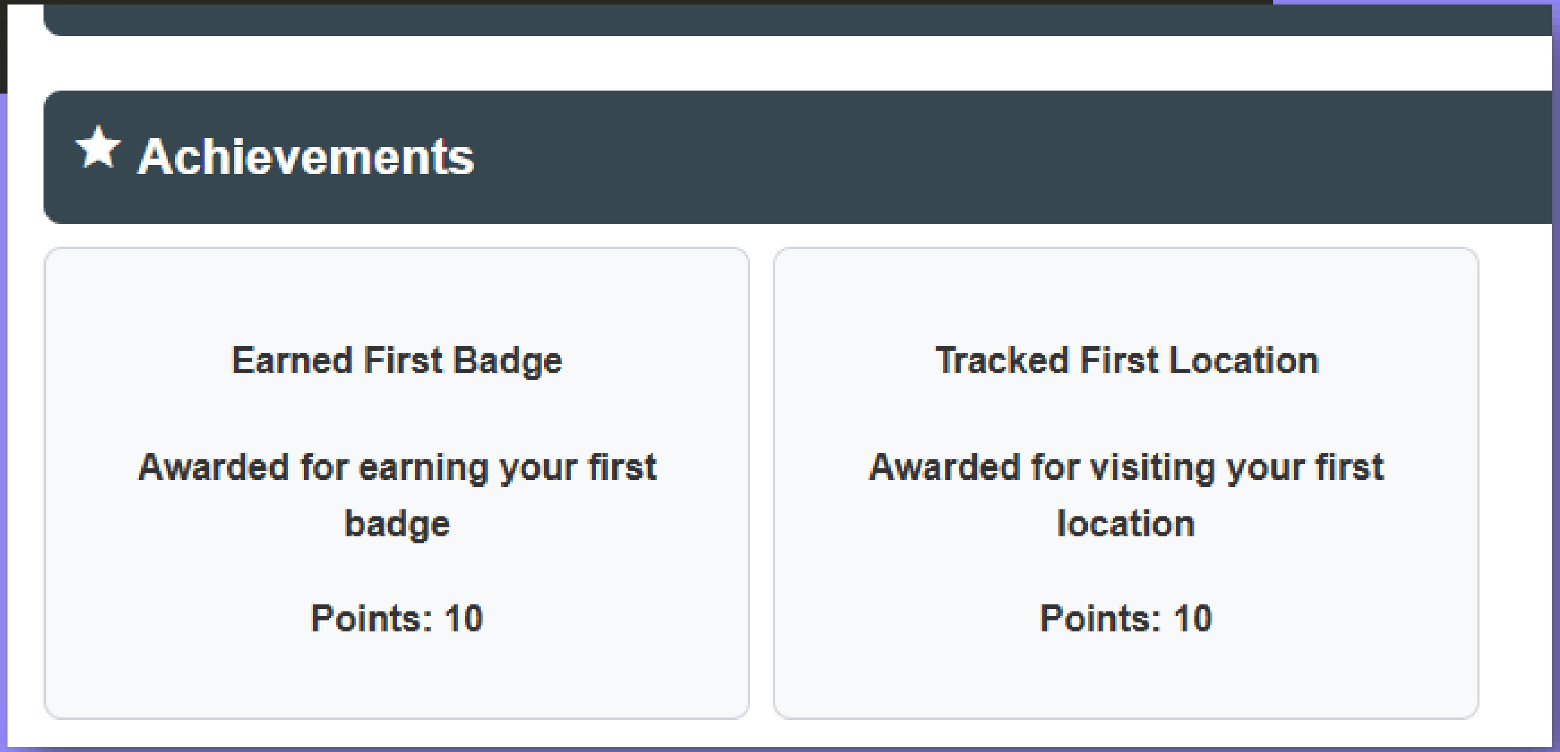
E

S

# Gamified UI Elements – Achievements

```
const showAchievementPopupMessage = (points, text) => {
  setAchievementMessage(`Congrats! You've earned ${points} points for ${text}`);
  setShowAchievementPopup(true);
  setTimeout(() => {
    setShowAchievementPopup(false);
  }, 3000);
};
```

```
{earnedAchievements.map((achievement, index) => (
  <div key={index} className="achievement-card">
    <h4>{achievement.text}</h4>
    <p>{achievement.description}</p>
    <p>Points: {achievement.points}</p>
  </div>
))}
```



★ Achievements

Earned First Badge

Awarded for earning your first badge

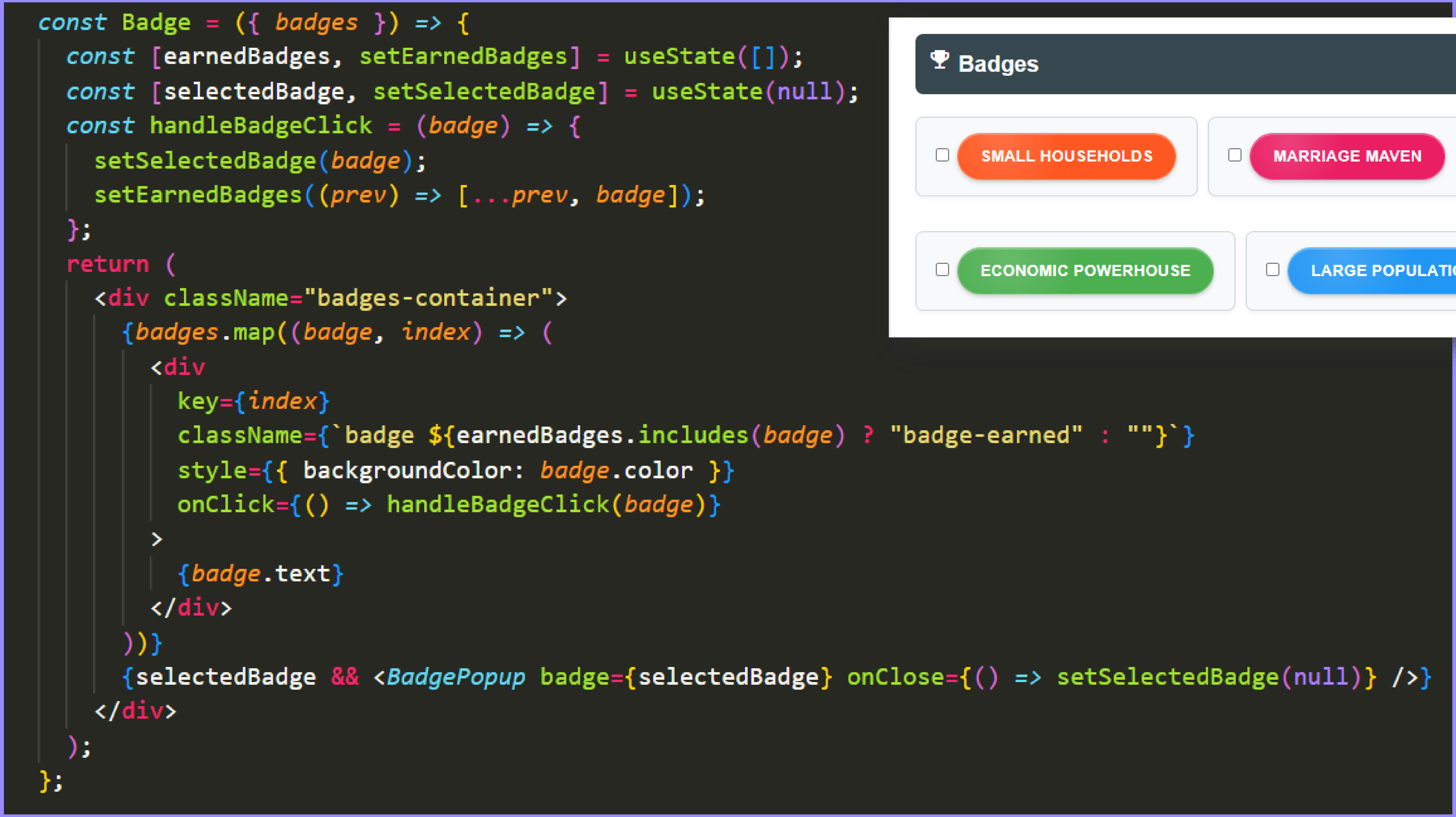
Points: 10

Tracked First Location

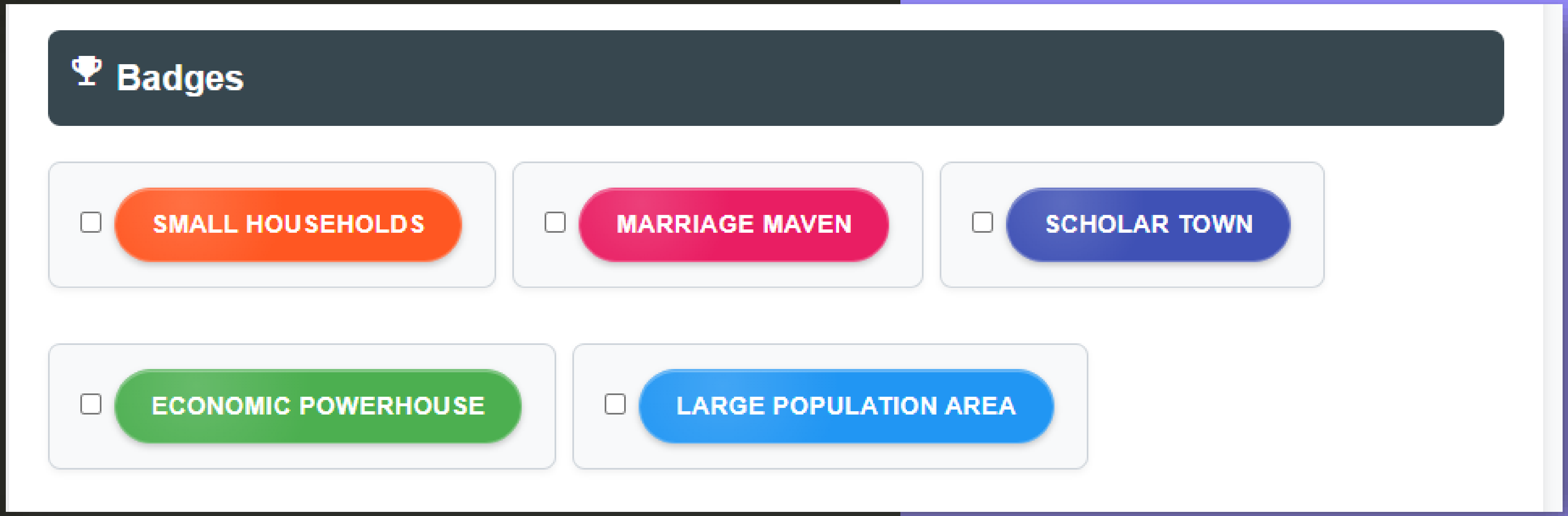
Awarded for visiting your first location

Points: 10

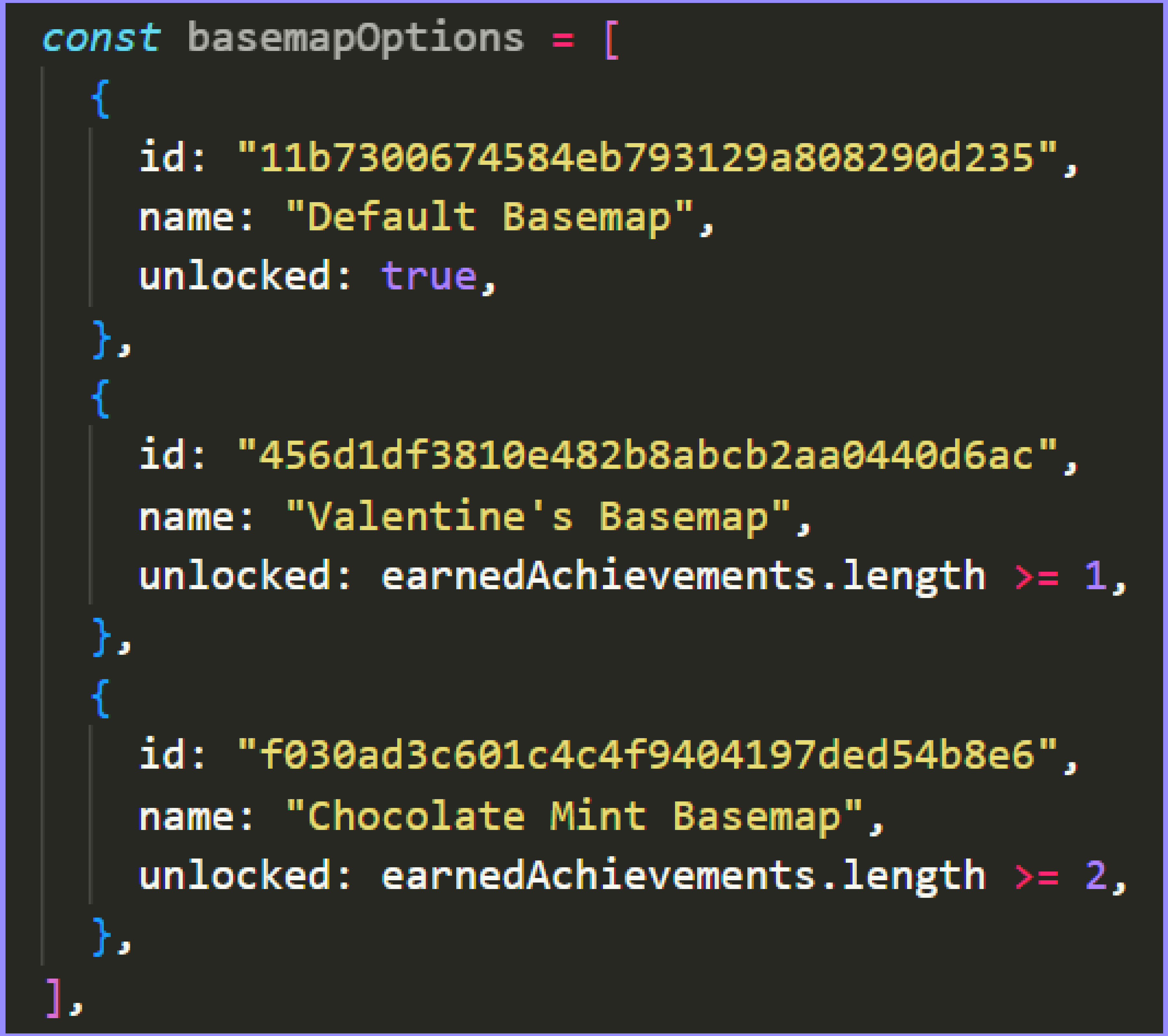
# Gamified UI Elements – Badge



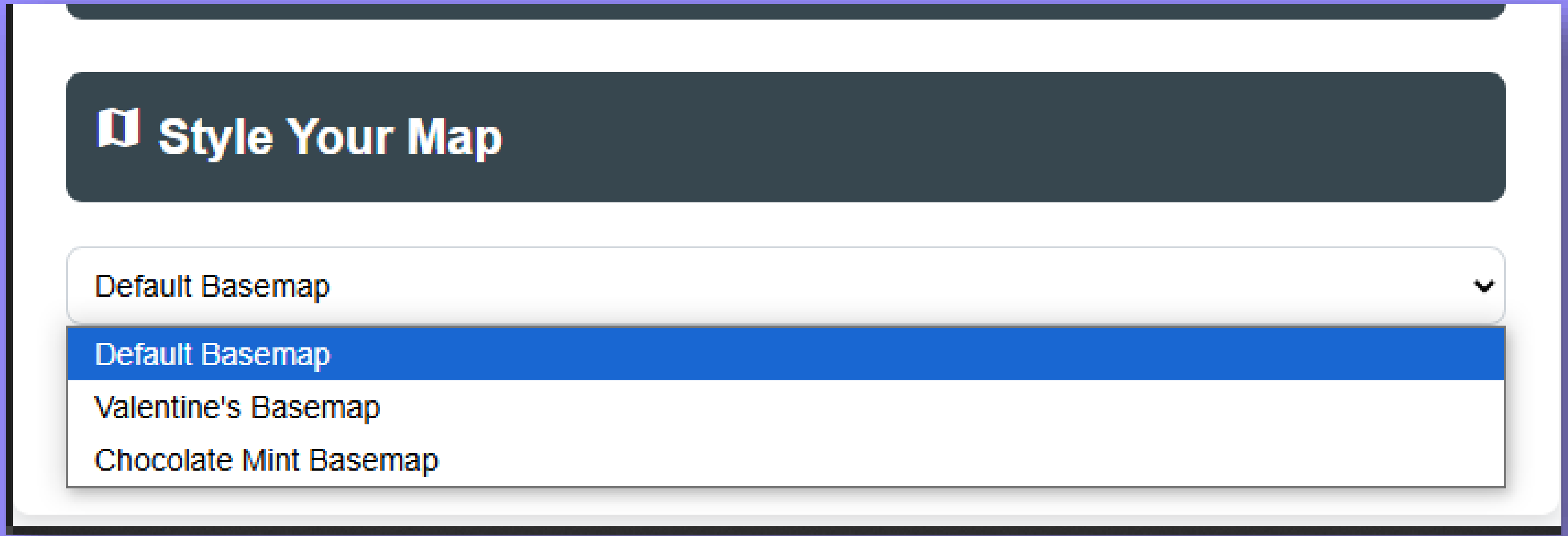
```
const Badge = ({ badges }) => {
  const [earnedBadges, setEarnedBadges] = useState([]);
  const [selectedBadge, setSelectedBadge] = useState(null);
  const handleBadgeClick = (badge) => {
    setSelectedBadge(badge);
    setEarnedBadges((prev) => [...prev, badge]);
  };
  return (
    <div className="badges-container">
      {badges.map((badge, index) => (
        <div
          key={index}
          className={`badge ${earnedBadges.includes(badge) ? "badge-earned" : ""}`}
          style={{ backgroundColor: badge.color }}
          onClick={() => handleBadgeClick(badge)}
        >
          {badge.text}
        </div>
      ))}
      {selectedBadge && <BadgePopup badge={selectedBadge} onClose={() => setSelectedBadge(null)} />}
    </div>
  );
};
```



# Gamified UI Elements – Basemap Options



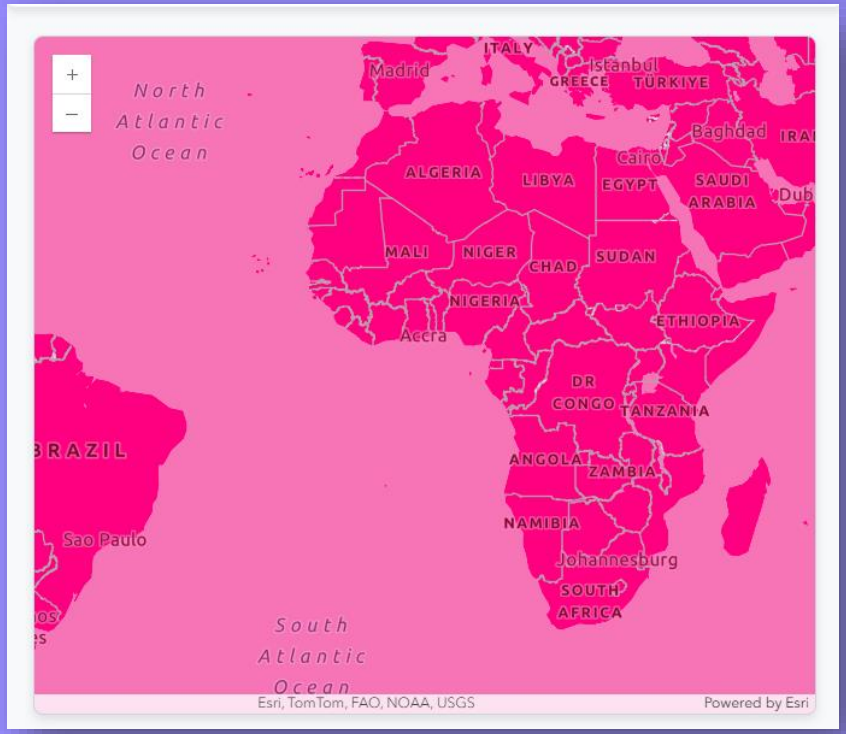
```
const basemapOptions = [
  {
    id: "11b7300674584eb793129a808290d235",
    name: "Default Basemap",
    unlocked: true,
  },
  {
    id: "456d1df3810e482b8abcb2aa0440d6ac",
    name: "Valentine's Basemap",
    unlocked: earnedAchievements.length >= 1,
  },
  {
    id: "f030ad3c601c4c4f9404197ded54b8e6",
    name: "Chocolate Mint Basemap",
    unlocked: earnedAchievements.length >= 2,
  },
],
```

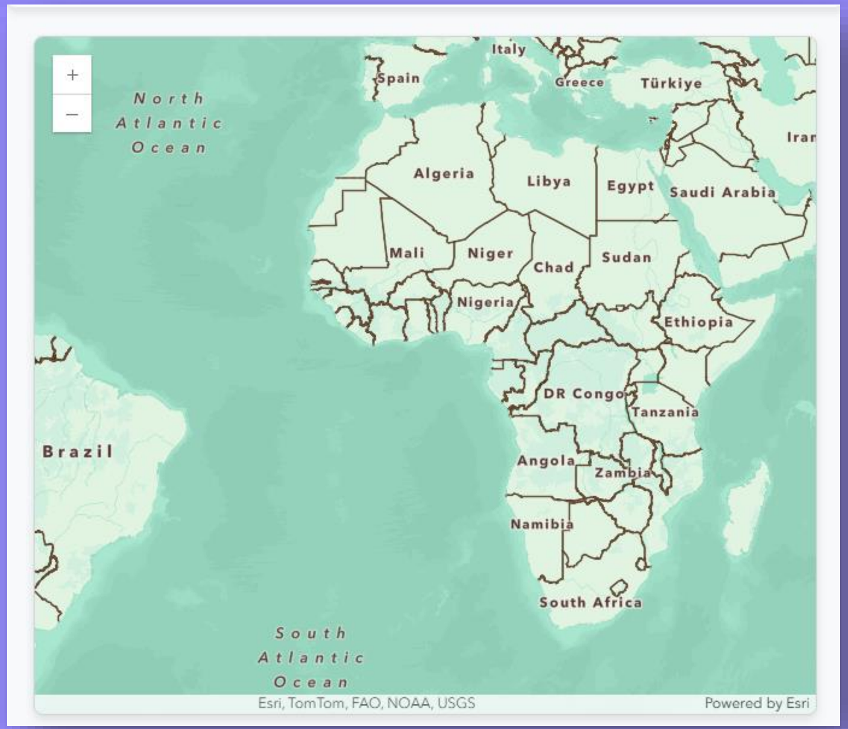


Style Your Map

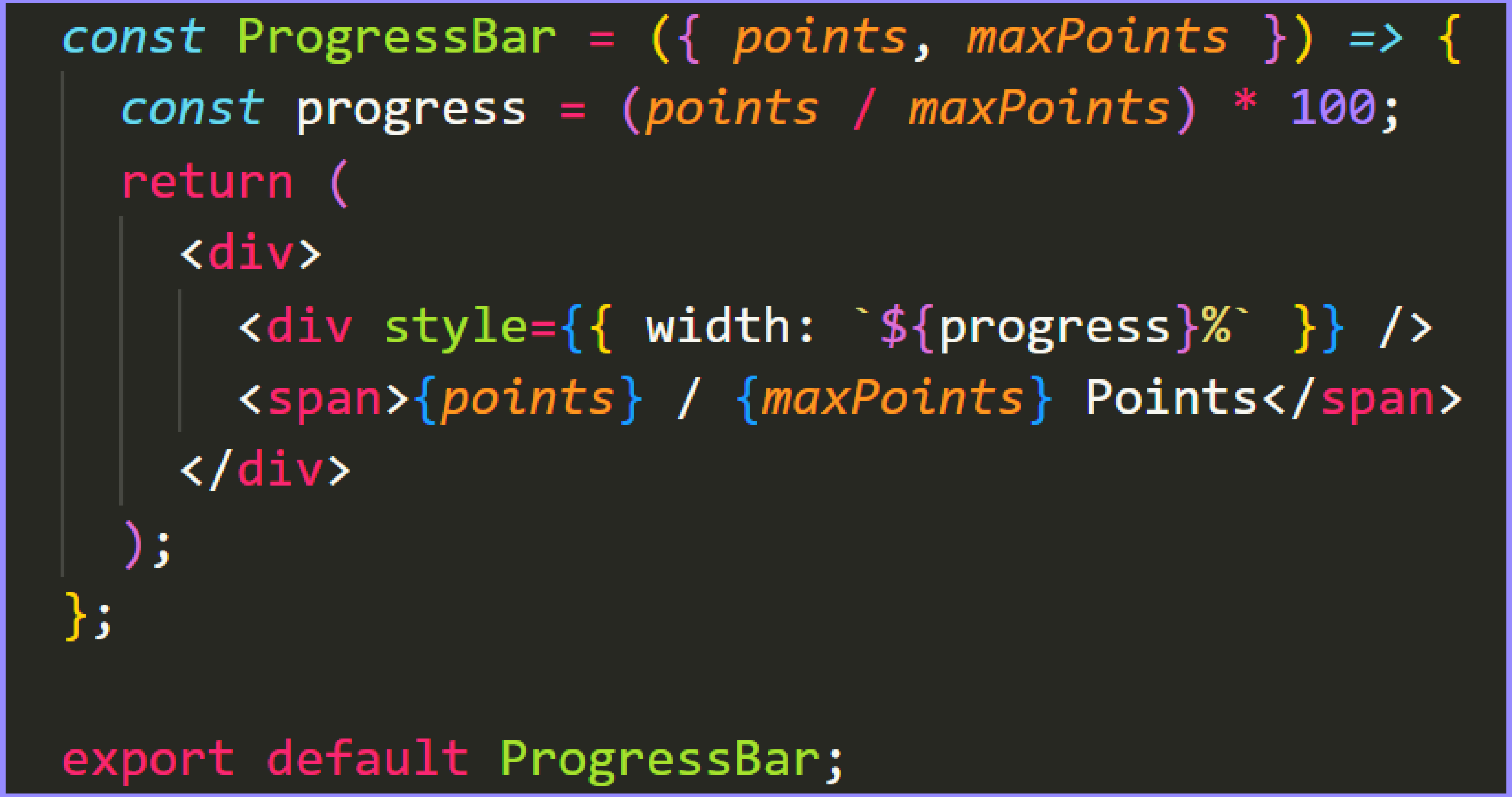
Default Basemap

Default Basemap
Valentine's Basemap
Chocolate Mint Basemap

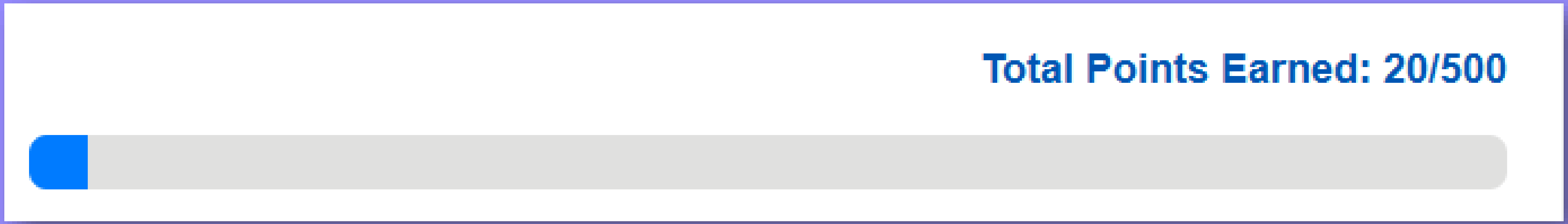




# Gamified UI Elements – Progress Bar



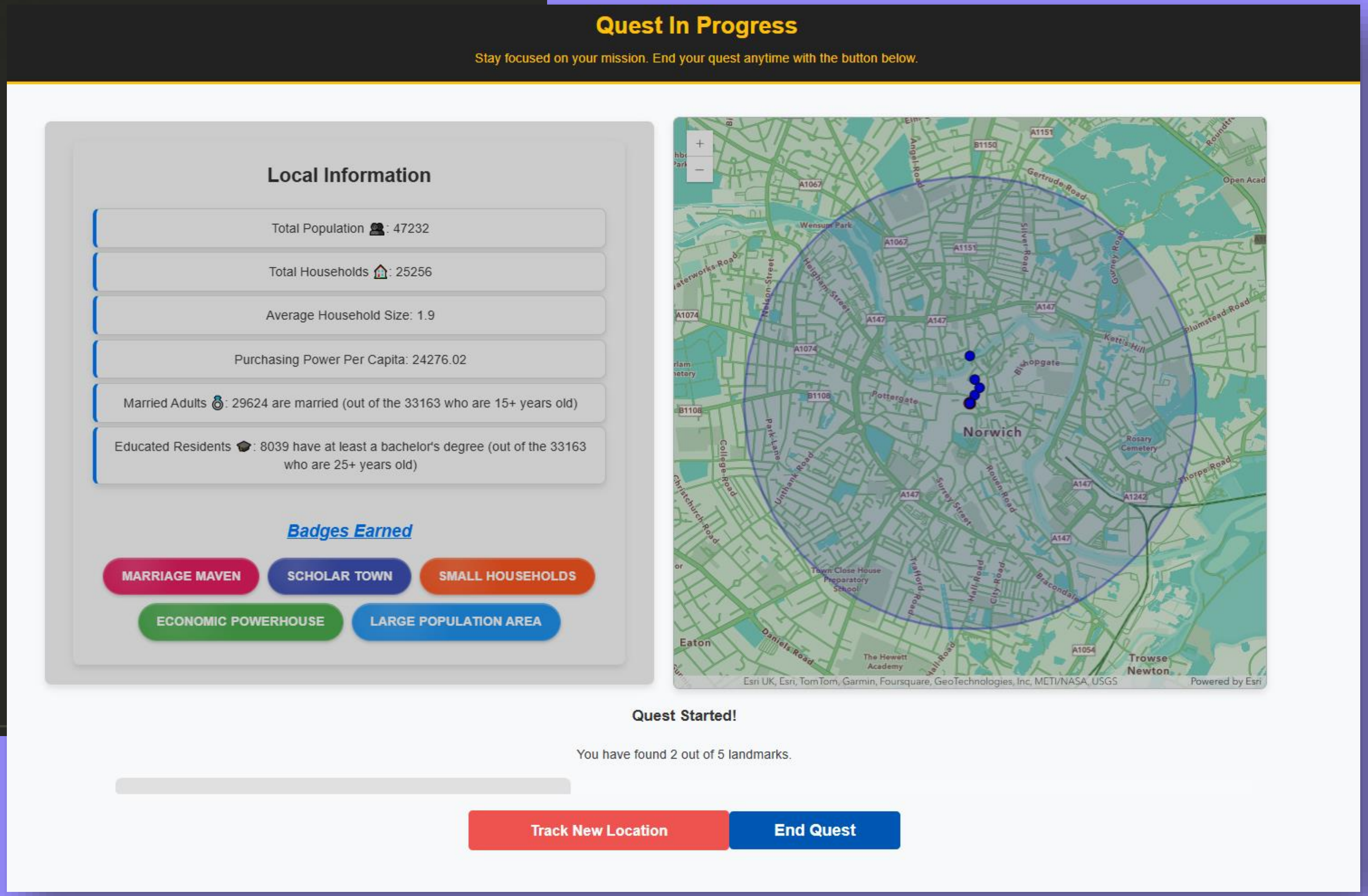
```
const ProgressBar = ({ points, maxPoints }) => {
  const progress = (points / maxPoints) * 100;
  return (
    <div>
      <div style={{ width: `${progress}%` }} />
      <span>{points} / {maxPoints} Points</span>
    </div>
  );
};

export default ProgressBar;
```



Total Points Earned: 20/500

# Gamified UI Elements – Quests

```
const QuestStatus = ({ questStarted, foundLandmarks, totalLandmarks }) => {
  if (!questStarted) return null;
  const progressPercentage = Math.round((foundLandmarks / totalLandmarks) * 100);
  return (
    <div className="quest-status-container">
      <h3>Quest Started!</h3>
      <p>
        You have found {foundLandmarks} out of {totalLandmarks} landmarks.
      </p>
      <div className="progress-bar-container">
        <div
          className="progress-bar"
          style={{ width: `${progressPercentage}%` }}
        ></div>
      </div>
    </div>
  );
};
```



**Quest In Progress**

Stay focused on your mission. End your quest anytime with the button below.

**Local Information**

Total Population 👥: 47232

Total Households 🏠: 25256

Average Household Size: 1.9

Purchasing Power Per Capita: 24276.02

Married Adults 💍: 29624 are married (out of the 33163 who are 15+ years old)

Educated Residents 🎓: 8039 have at least a bachelor's degree (out of the 33163 who are 25+ years old)

***Badges Earned***

**Quest Started!**

You have found 2 out of 5 landmarks.

G

**G**amified UI Elements

A

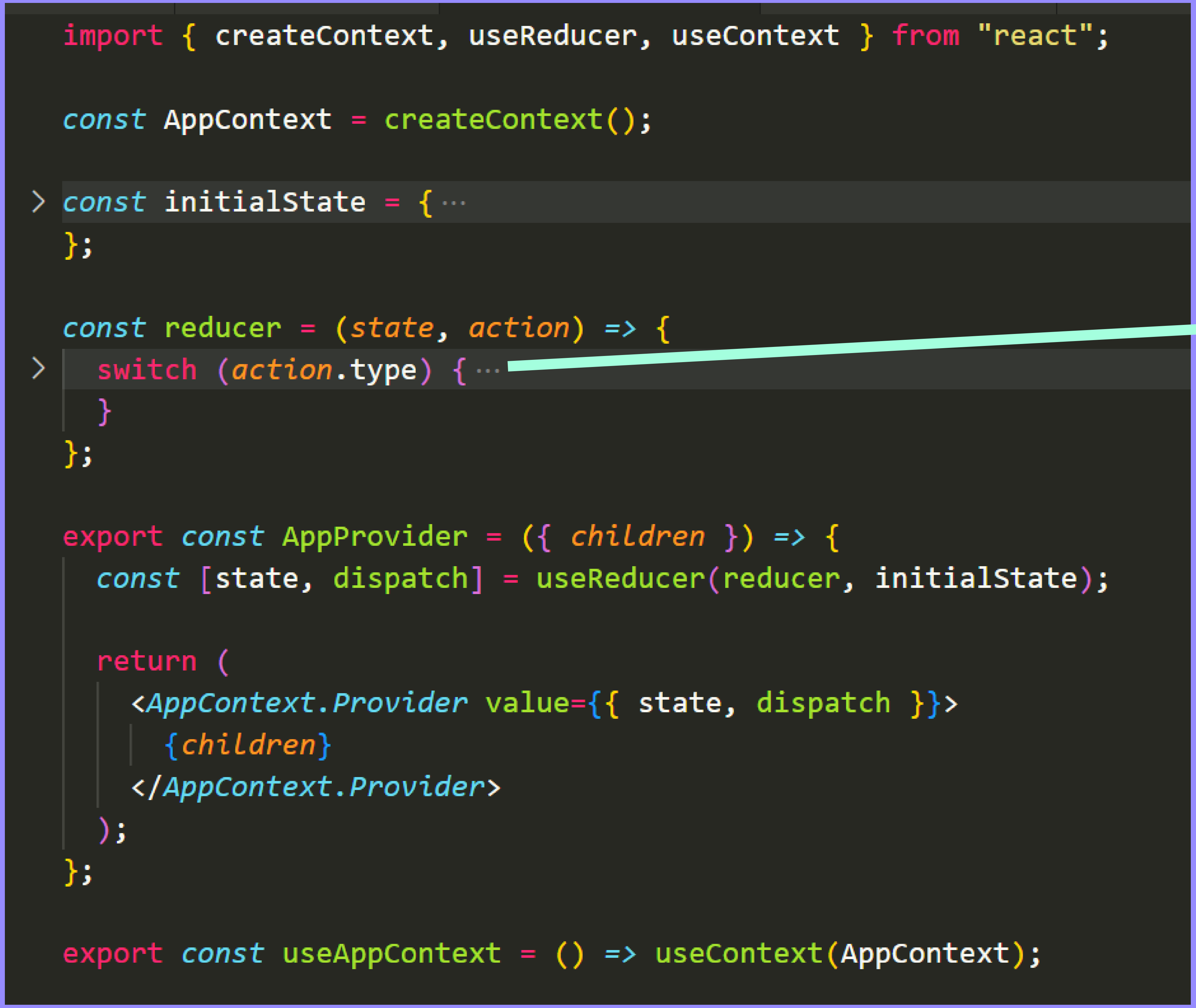
**A**dvanced State Control

M

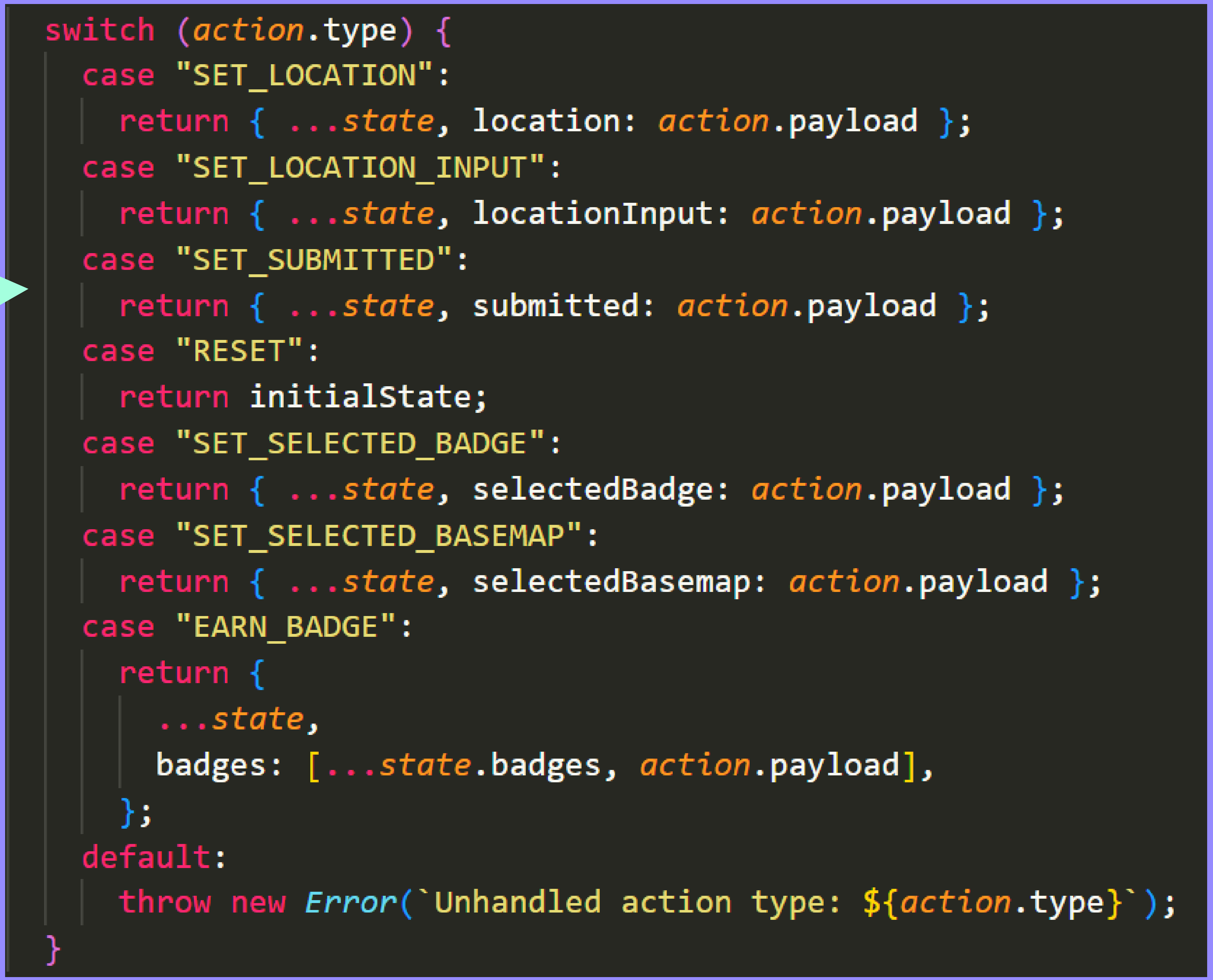
E

S

# Advanced State Control – Context and Reducer

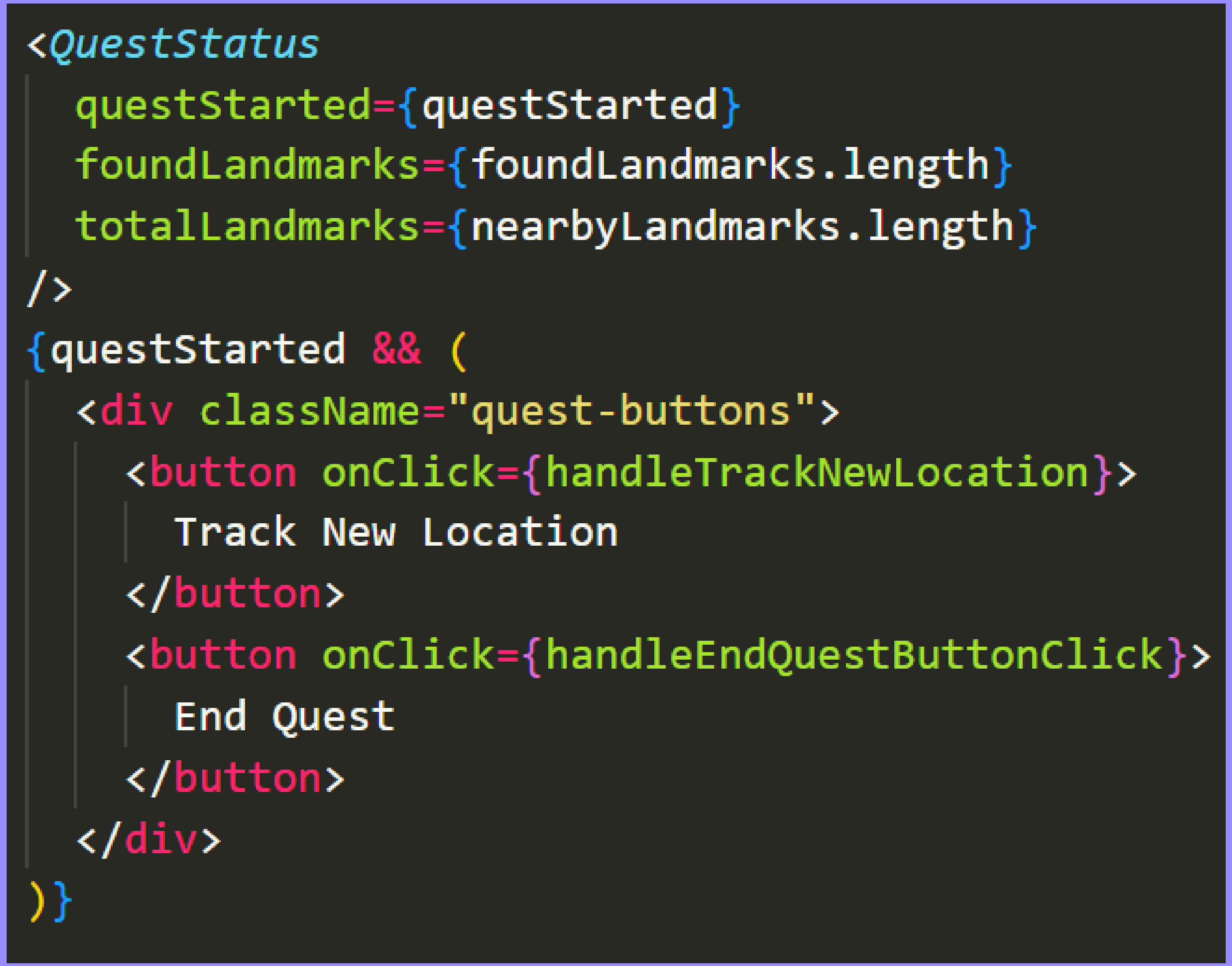


```
import { createContext, useReducer, useContext } from "react";

const AppContext = createContext();

const initialState = {⋯
};

const reducer = (state, action) => {
  switch (action.type) {⋯
  }
};

export const AppProvider = ({ children }) => {
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <AppContext.Provider value={{ state, dispatch }}>
      {children}
    </AppContext.Provider>
  );
};

export const useAppContext = () => useContext(AppContext);
```

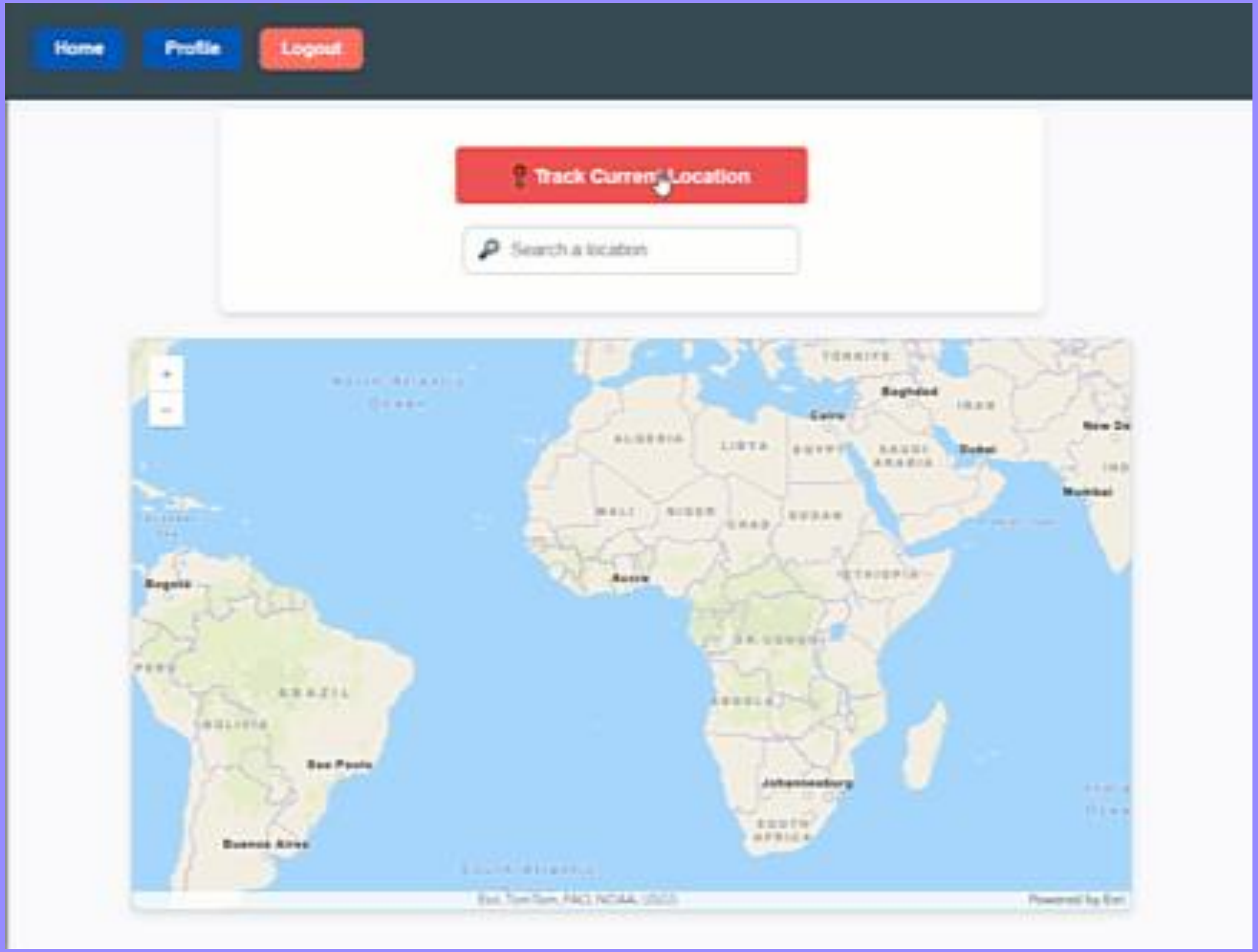


```
switch (action.type) {
  case "SET_LOCATION":
    return { ...state, location: action.payload };
  case "SET_LOCATION_INPUT":
    return { ...state, locationInput: action.payload };
  case "SET_SUBMITTED":
    return { ...state, submitted: action.payload };
  case "RESET":
    return initialState;
  case "SET_SELECTED_BADGE":
    return { ...state, selectedBadge: action.payload };
  case "SET_SELECTED_BASEMAP":
    return { ...state, selectedBasemap: action.payload };
  case "EARN_BADGE":
    return {
      ...state,
      badges: [...state.badges, action.payload],
    };
  default:
    throw new Error(`Unhandled action type: ${action.type}`);
}
```

# Advanced State Control – Quest View



```jsx
<QuestStatus
  questStarted={questStarted}
  foundLandmarks={foundLandmarks.length}
  totalLandmarks={nearbyLandmarks.length}
/>
{questStarted && (
  <div className="quest-buttons">
    <button onClick={handleTrackNewLocation}>
      Track New Location
    </button>
    <button onClick={handleEndQuestButtonClick}>
      End Quest
    </button>
  </div>
)}
```



**G**

**G**amified UI Elements

**A**

**A**dvanced State Control

**M**

**M**emoization

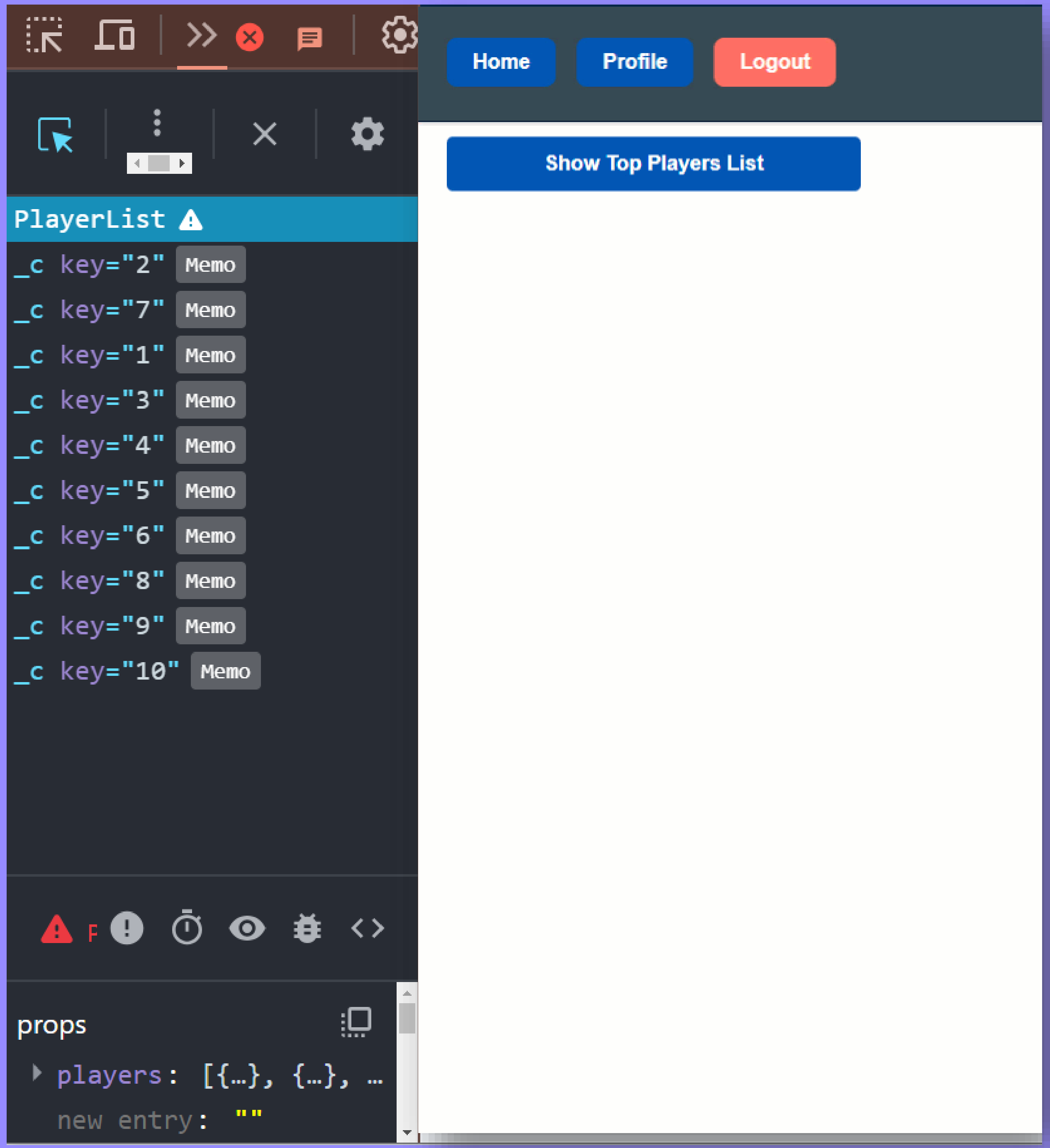
**E**

**S**

# Memoization – Players List

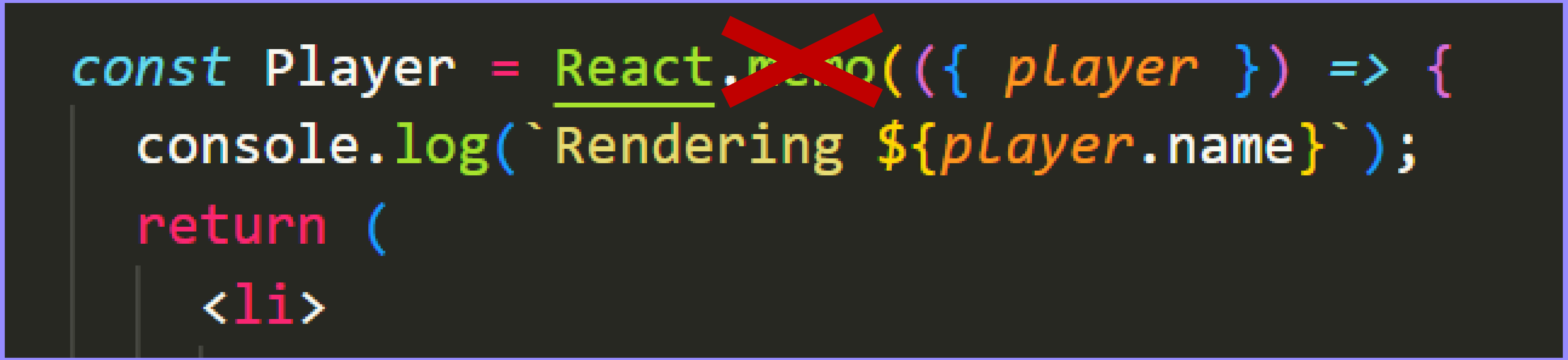
```
import React from 'react';

const Player = React.memo(({ player }) => {
  console.log(`Rendering ${player.name}`);
  return (
    <li>
      {player.name}: {player.score}
    </li>
  );
});

export default Player;
```

```
import { useMemo } from 'react';
import Player from './Player';

const PlayerList = ({ players }) => {
  const sortedPlayers = useMemo(() => {
    return players.sort((a, b) => b.score - a.score);
  }, [players]);

  return (
    <div className="player-list">
      <h3>Top Players</h3>
      <ul>
        {sortedPlayers.map(player => (
          <Player key={player.id} player={player} />
        ))}
      </ul>
    </div>
  );
};

export default PlayerList;
```

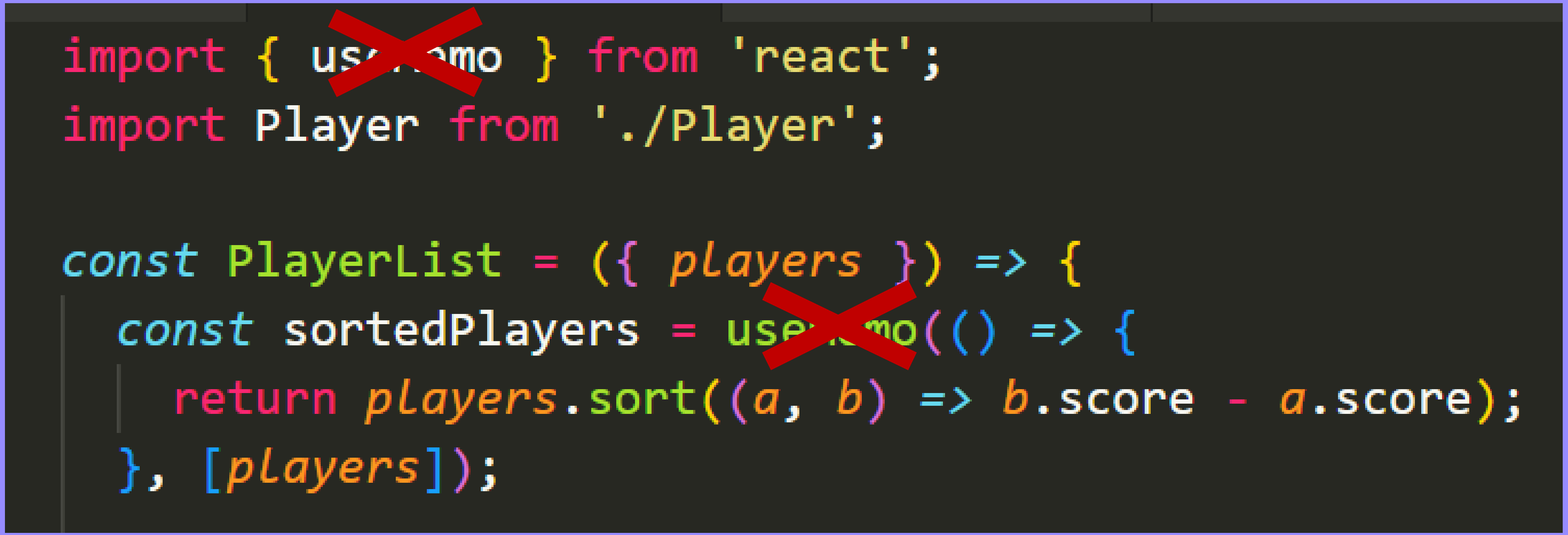


```
PlayerList ⚠
_c key="2" Memo
_c key="7" Memo
_c key="1" Memo
_c key="3" Memo
_c key="4" Memo
_c key="5" Memo
_c key="6" Memo
_c key="8" Memo
_c key="9" Memo
_c key="10" Memo

props
▸ players: [{…}, {…}, …
  new entry: ""
```

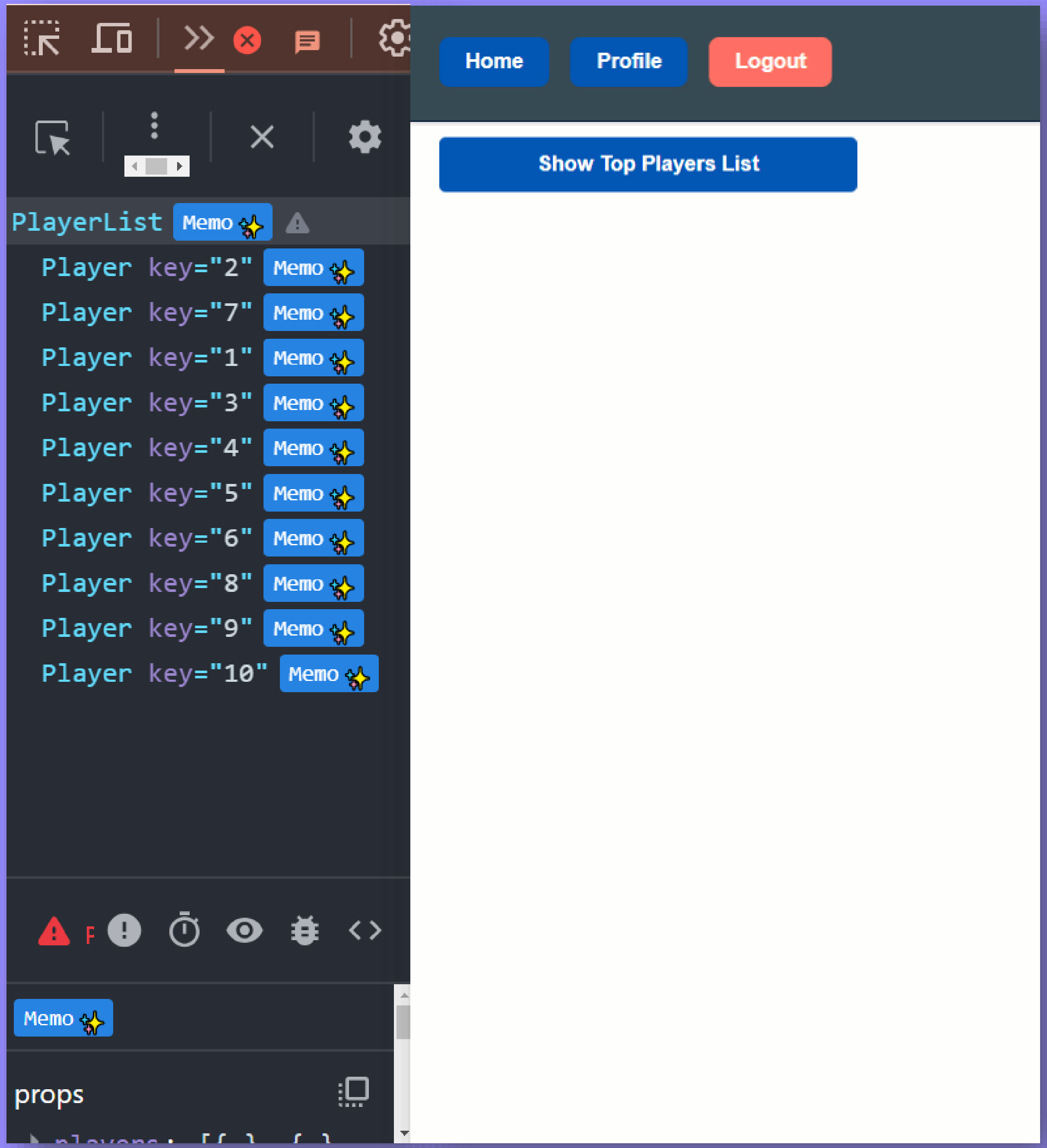
# Memoization – Players List with React Compiler



```
const Player = React.~~memo~~(({ player }) => {
  console.log(`Rendering ${player.name}`);
  return (
    <li>
```



```
import { ~~useMemo~~ } from 'react';
import Player from './Player';

const PlayerList = ({ players }) => {
  const sortedPlayers = ~~useMemo~~(() => {
    return players.sort((a, b) => b.score - a.score);
  }, [players]);
```



```
PlayerList Memo ✨
  Player key="2" Memo ✨
  Player key="7" Memo ✨
  Player key="1" Memo ✨
  Player key="3" Memo ✨
  Player key="4" Memo ✨
  Player key="5" Memo ✨
  Player key="6" Memo ✨
  Player key="8" Memo ✨
  Player key="9" Memo ✨
  Player key="10" Memo ✨

Memo ✨
props
```

**G**

**G**amified UI Elements

**A**

**A**dvanced State Control

**M**

**M**emoization

**E**

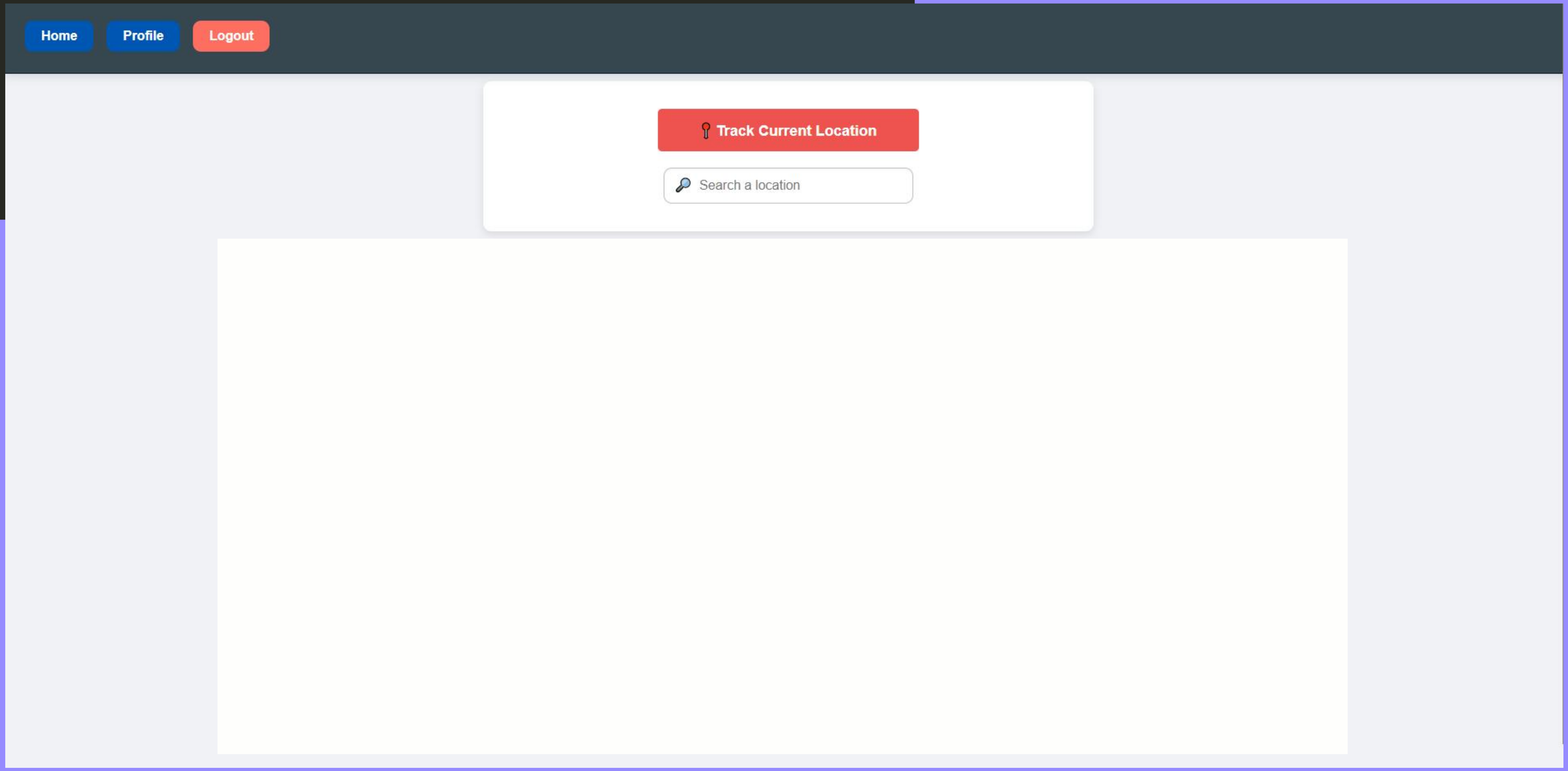
**E**fficient Rendering

**S**

# Efficient Rendering – Lazy and Suspense

```
const MapViewComponent = lazy(() => import("./components/MapViewComponent"));
const SimpleMapComponent = lazy(() => import("./components/SimpleMapComponent"));
```

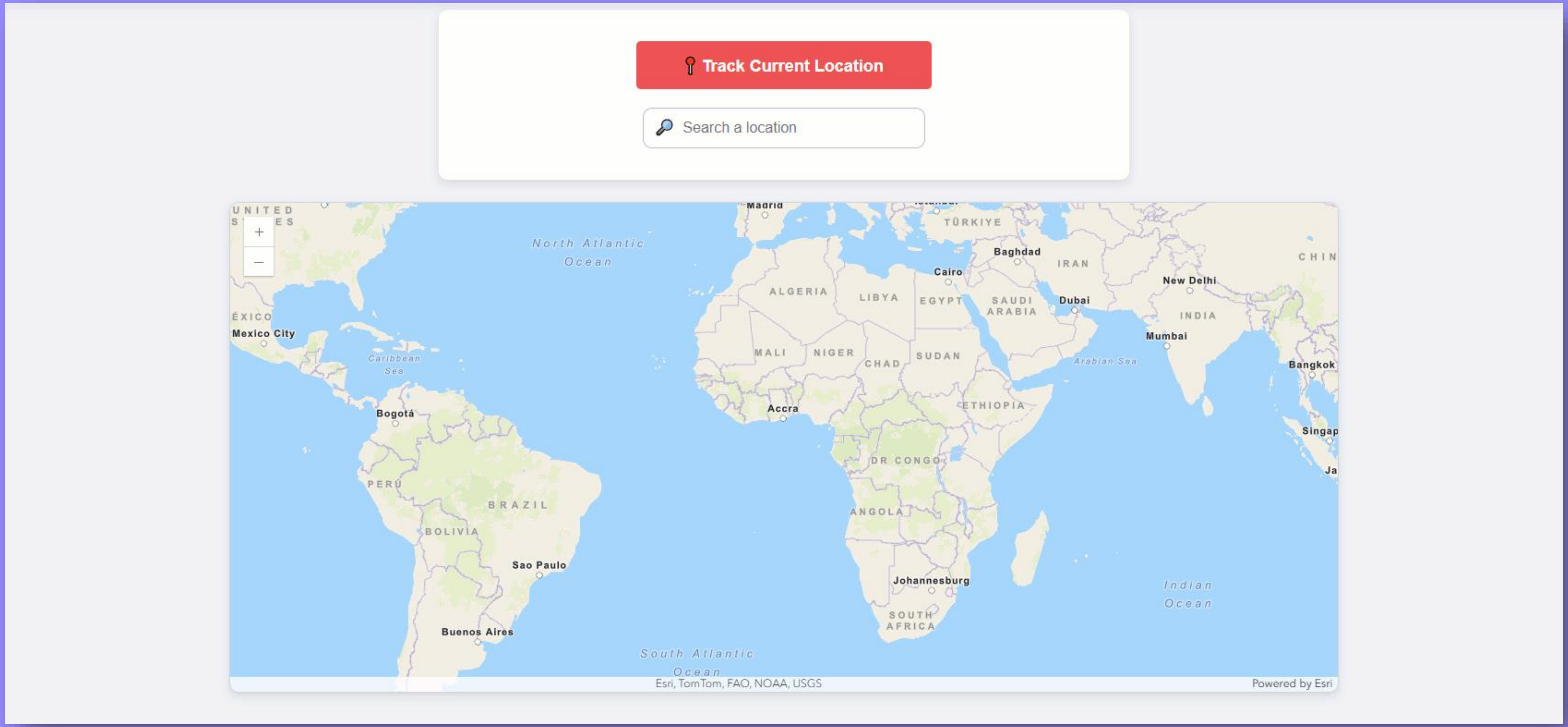
```
<Suspense fallback={<div>Loading Map...</div>}>
  {state.location ? (
    <MapViewComponent location={state.location} landmarks={nearbyLandmarks} />
  ) : (
    <SimpleMapComponent  />
  )}
</Suspense>
```



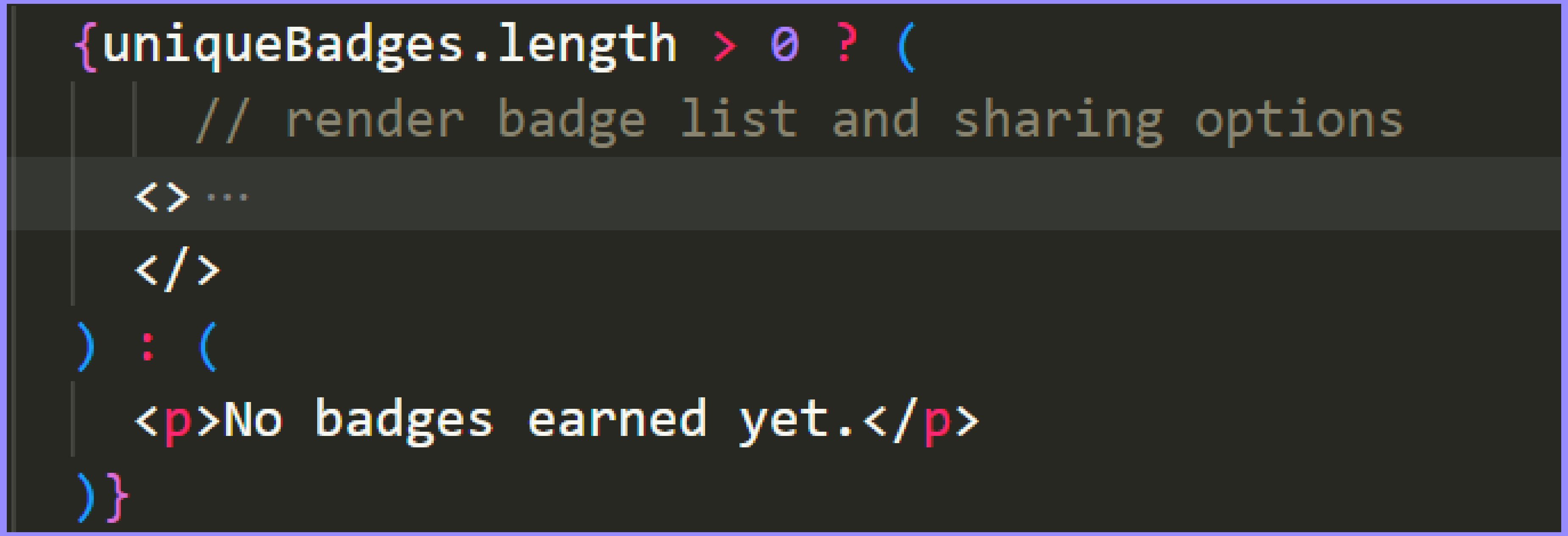
# Efficient Rendering – useEffect and useRef

```
const mapDiv = useRef(null);
```

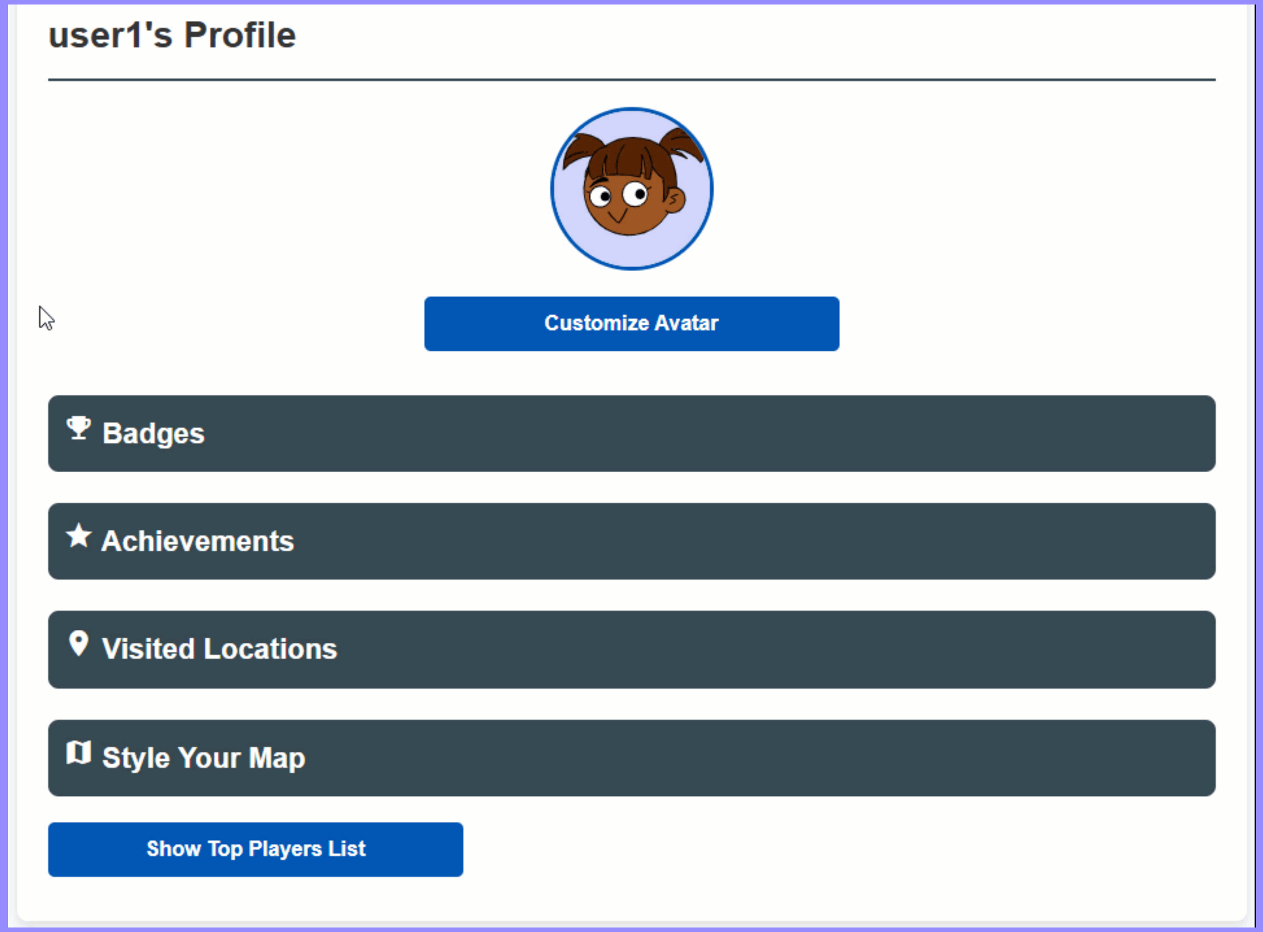
```
useEffect(() => {
  if (!location || !mapDiv.current) return;
  // Map initialization code...
  return () => view.destroy();
}, [location, landmarks, selectedBasemap]);
```



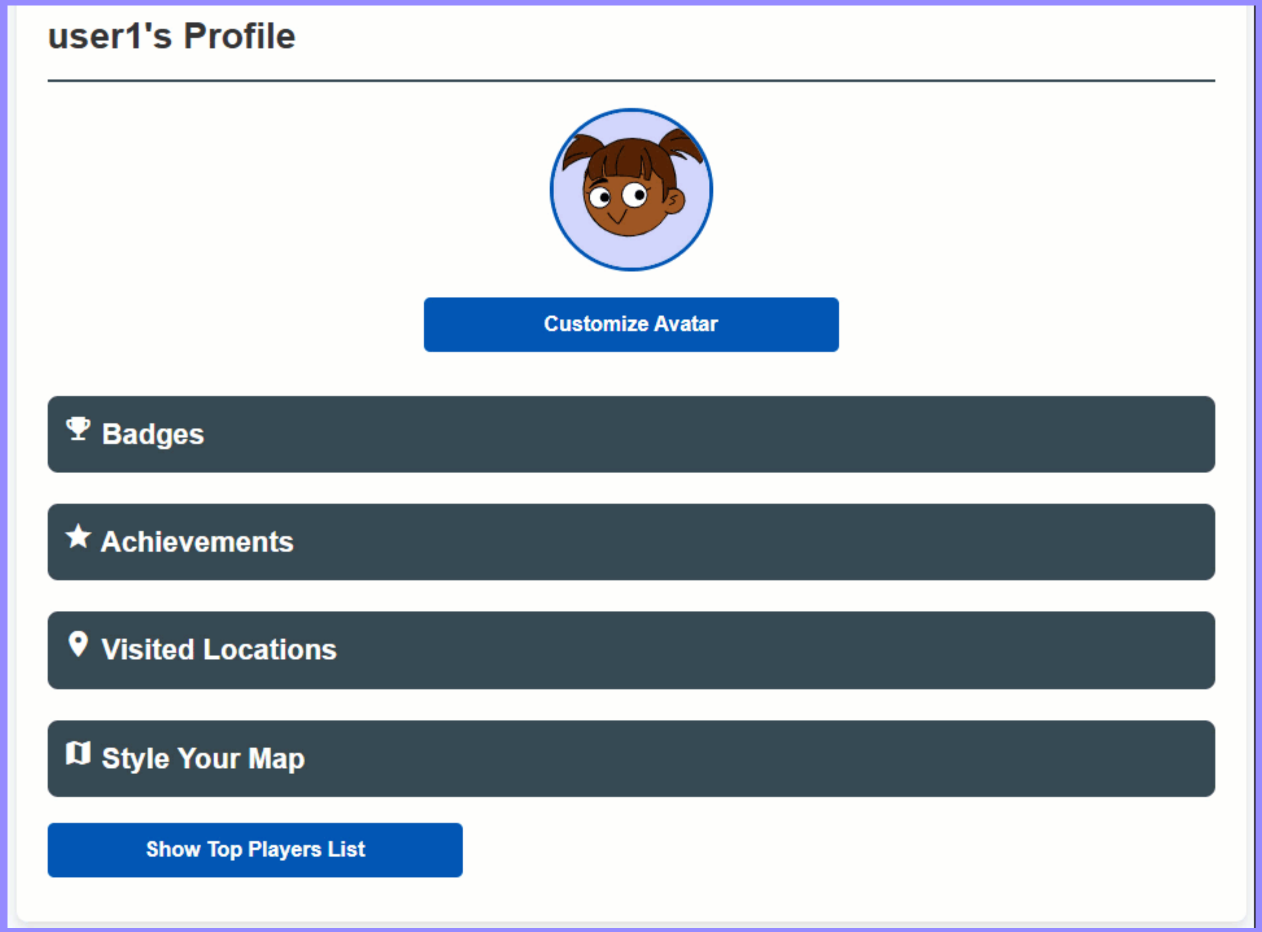
# Efficient Rendering – On-Demand Badge Rendering



```
{uniqueBadges.length > 0 ? (
    // render badge list and sharing options
  <> ···
  </>
) : (
  <p>No badges earned yet.</p>
)}
```



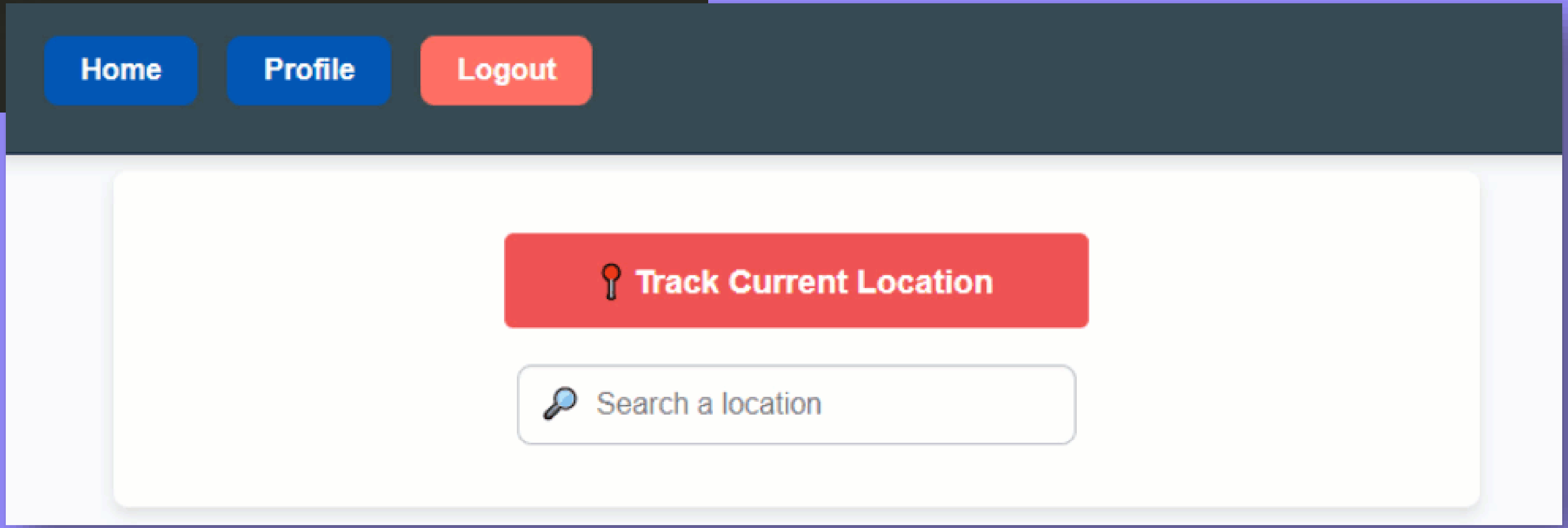
user1's Profile



user1's Profile

## Efficient Rendering – On-Demand Achievement Popups

```
const queueAchievementPopup = (points, text) => {
  setAchievementQueue((prevQueue) => [...prevQueue, { points, text }]);
};

useEffect(() => {
  if (achievementQueue.length > 0 && !showAchievementPopup) {
    const { points, text } = achievementQueue[0];
    showAchievementPopupMessage(points, text);
    setAchievementQueue((prevQueue) => prevQueue.slice(1));
  }
}, [achievementQueue, showAchievementPopup]);
```



G

**G**amified UI Elements

A

**A**dvanced State Control

M

**M**emoization

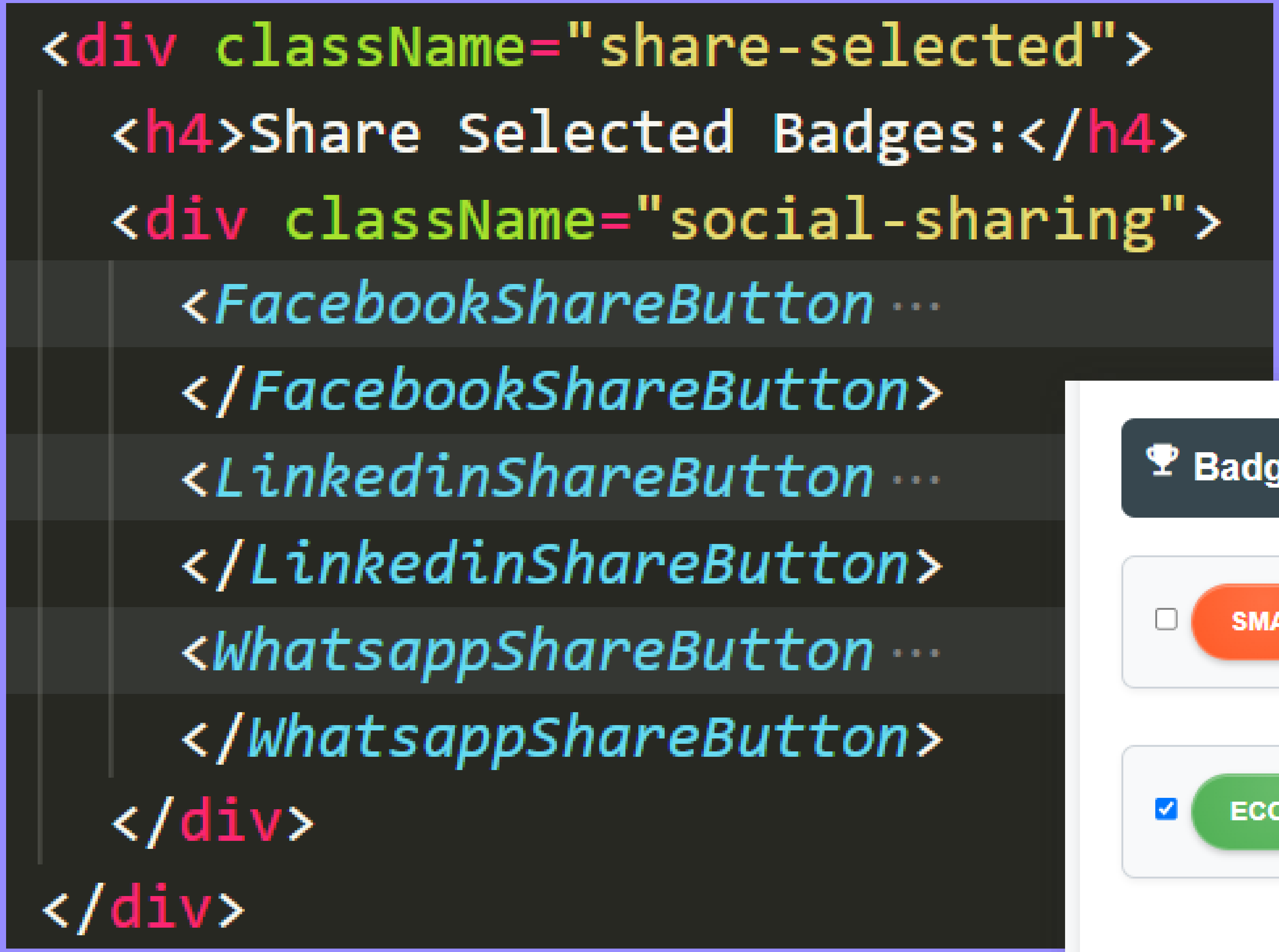
E

**E**fficient Rendering

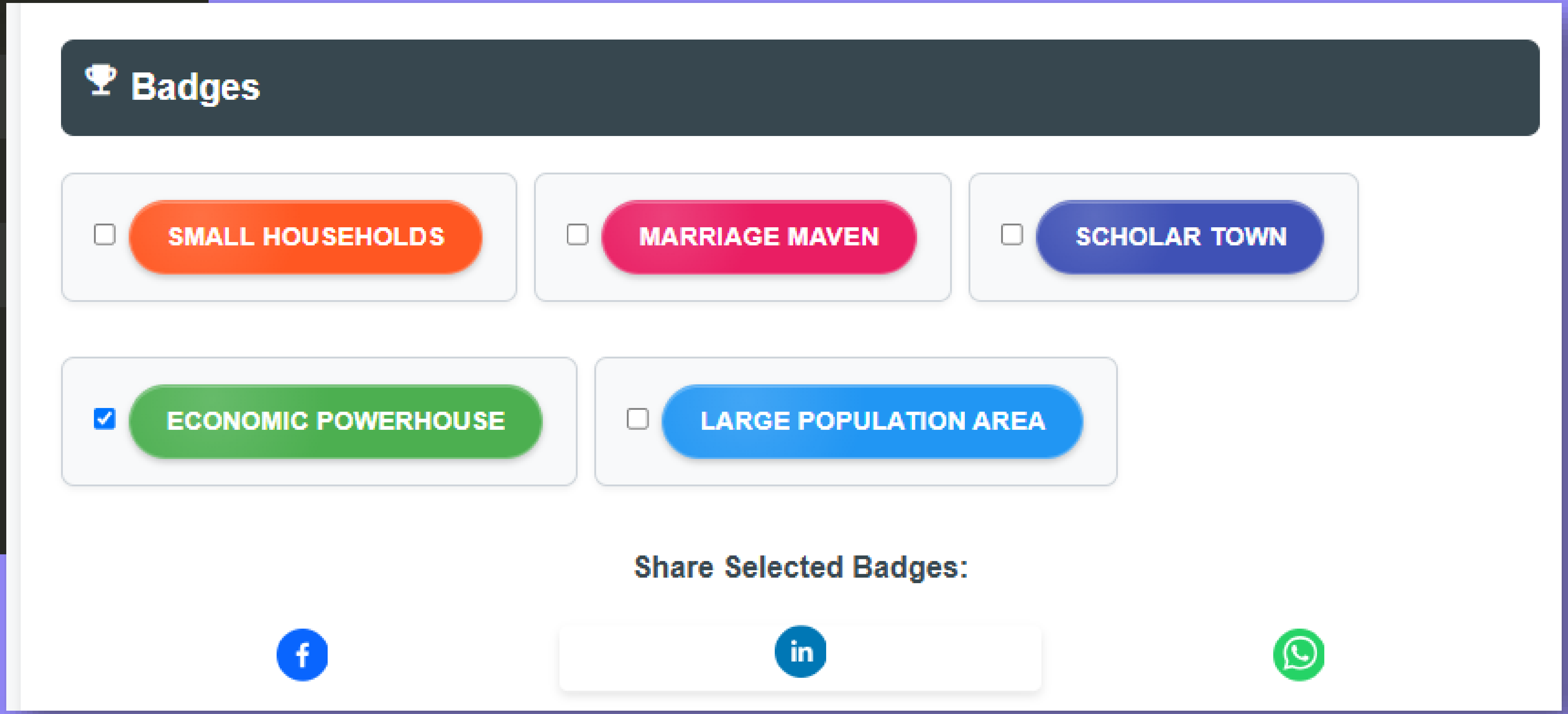
S

**S**ocial Interaction

## Social Interaction – Share Badges



```
<div className="share-selected">
  <h4>Share Selected Badges:</h4>
  <div className="social-sharing">
    <FacebookShareButton ···
    </FacebookShareButton>
    <LinkedinShareButton ···
    </LinkedinShareButton>
    <WhatsappShareButton ···
    </WhatsappShareButton>
  </div>
</div>
```



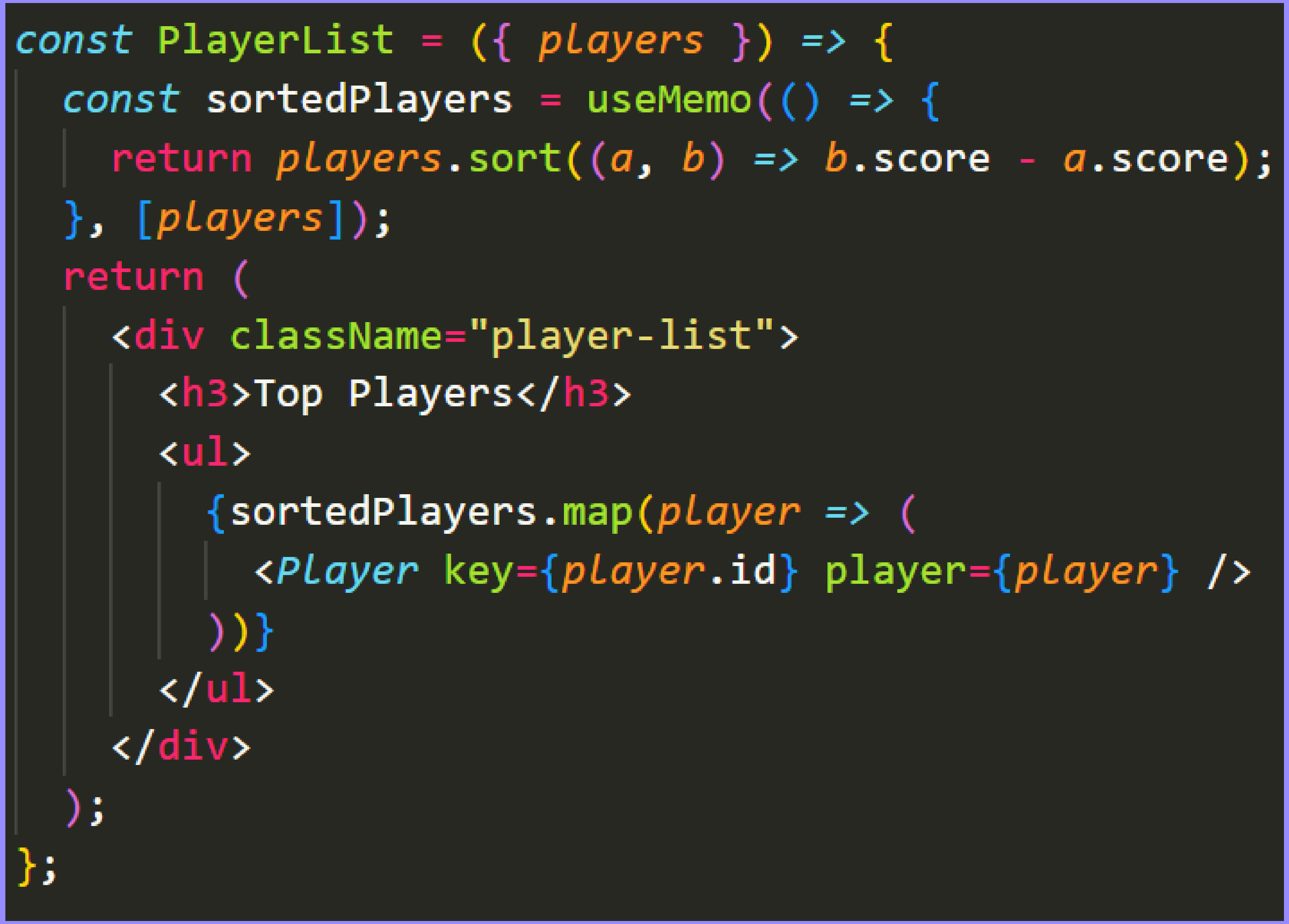
Badges

☐ SMALL HOUSEHOLDS ☐ MARRIAGE MAVEN ☐ SCHOLAR TOWN

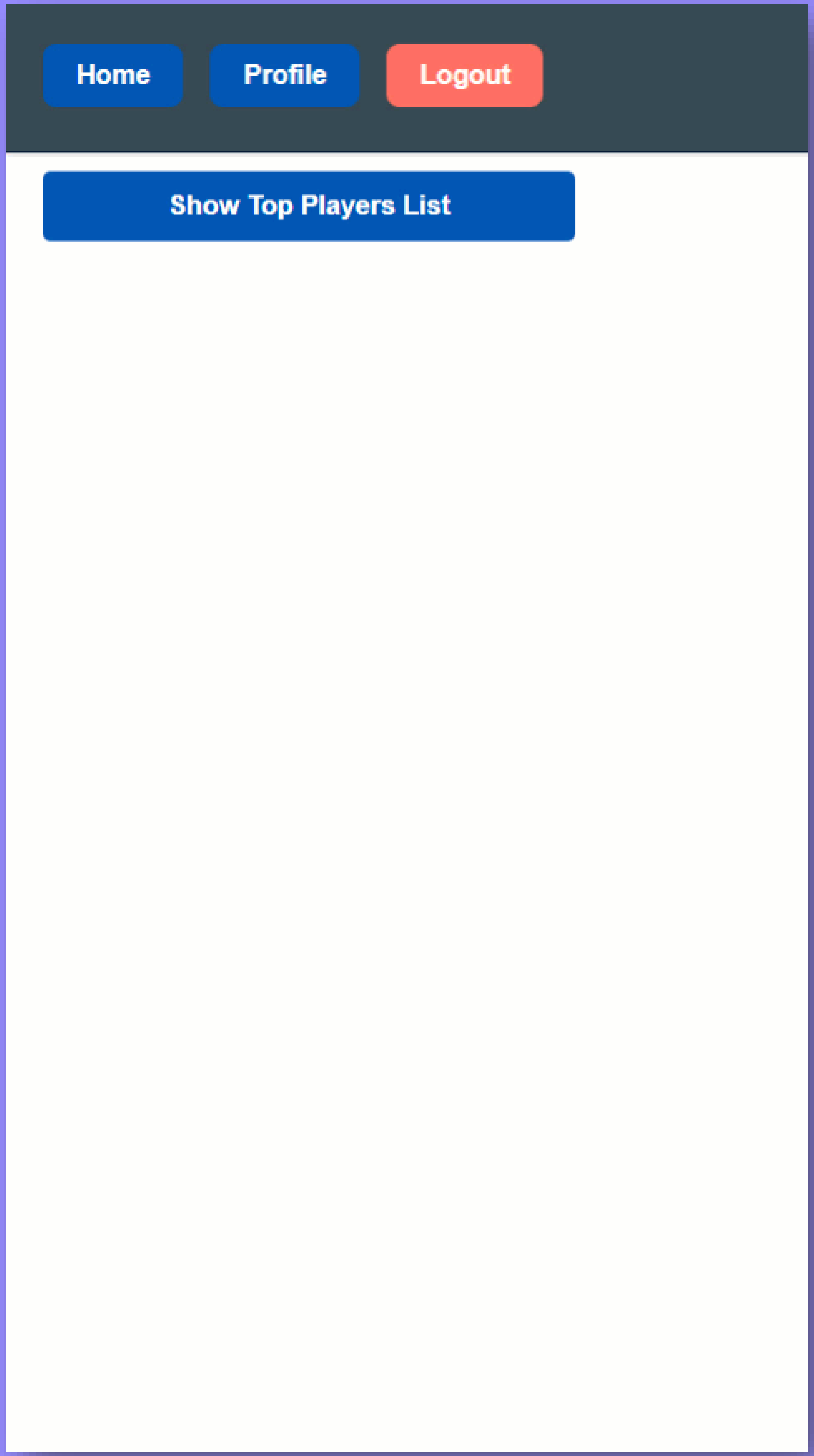
☑ ECONOMIC POWERHOUSE ☐ LARGE POPULATION AREA

Share Selected Badges:

## Social Interaction – Player List



```
const PlayerList = ({ players }) => {
  const sortedPlayers = useMemo(() => {
    return players.sort((a, b) => b.score - a.score);
  }, [players]);
  return (
    <div className="player-list">
      <h3>Top Players</h3>
      <ul>
        {sortedPlayers.map(player => (
          <Player key={player.id} player={player} />
        ))}
      </ul>
    </div>
  );
};
```



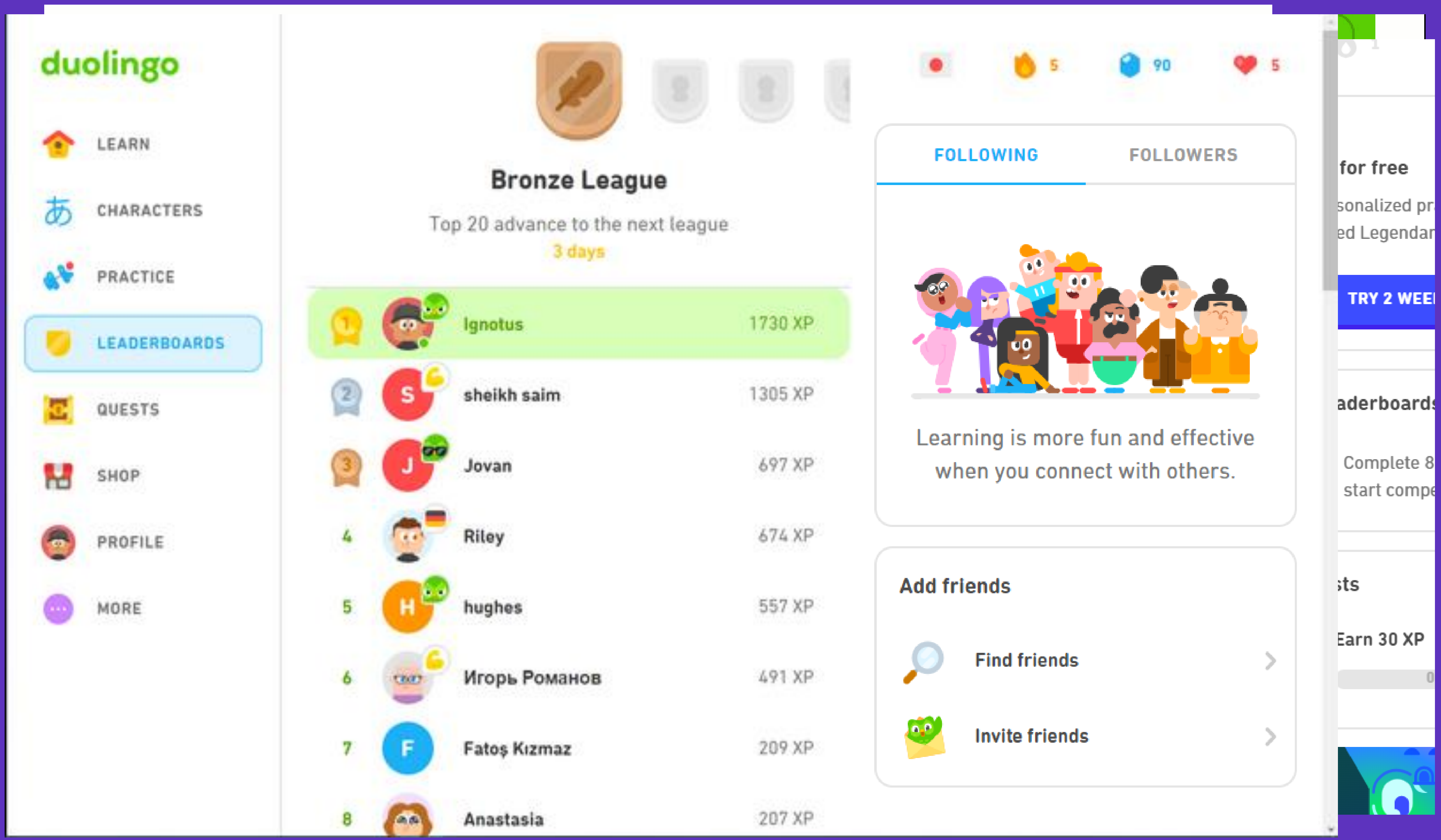
**G** Streak Counters & XP Bars

**A** Adaptive Learning System

**M** Optimized Lesson Rendering

**E** Efficient Component Rendering

**S** Leaderboards & Challenges


duolingo

LEARN
CHARACTERS
PRACTICE
LEADERBOARDS
QUESTS
SHOP
PROFILE
MORE

Bronze League

Top 20 advance to the next league

3 days

| | | |
|---|---|---|
| 1 | Ignotus | 1730 XP |
| 2 | sheikh saim | 1305 XP |
| 3 | Jovan | 697 XP |
| 4 | Riley | 674 XP |
| 5 | hughes | 557 XP |
| 6 | Игорь Романов | 491 XP |
| 7 | Fatoş Kızmaz | 209 XP |
| 8 | Anastasia | 207 XP |

5 90 5

FOLLOWING FOLLOWERS

Learning is more fun and effective when you connect with others.

Add friends

Find friends

Invite friends

for free

sonalized pr

ed Legendar

TRY 2 WEE

aderboard

Complete 8

start compe

sts

Earn 30 XP

# GAMES